Vysoká škola báňská – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra telekomunikační techniky

Připojení přístupových čipů Maxim iButton k procesoru Microchip

Connecting Approach Chips Maxim iButton to Processor Microchip

Diplomová práce

2018

Bc. Jiří Knapil

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Jiří Knapil**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Připojení přístupových čipů Maxim iButton k procesoru Microchip
Connecting Approach Chips Maxim iButton to Processor Microchip

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Proveďte stručnou rešerši kontaktních přístupových systémů.
2. Navrhněte přístupový systém se 4 čtečkami čipů Maxim iButton.
3. Vytvořte aplikaci pro osobní počítač v programovém nástroji Prometic podle pokynů vedoucího práce.
4. Předveďte provozuschopnost celého systému, realizujte ho ve formě funkčního vzoru.
5. K realizovanému systému vytvořte výukový materiál podle pokynů vedoucího práce.
6. Vytvořte úplnou technickou dokumentaci systému.

Seznam doporučené odborné literatury:

[1] Hindman, B. : APPLICATION NOTE 3808. Dostupné na https://www.maximintegrated.com/en/app-notes/index.mvp/id/3808
[2] Informační materiály firmy Maxim, dostupné na https://www.maximintegrated.com

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radek Novák, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 30.04.2018

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *29. června 2018*

podpis studenta

Poděkování

Rád bych poděkoval Ing. Radkovi Novákovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Diplomová práce se zabývá návrhem a realizací přístupového systému se 4 čtečkami čipů Maxim iButton. Cílem této práce bylo vytvořit kontaktní přístupový systém založený na platformě jednočipových mikropočítačů PIC Microchip.

Teoretická část práce je věnována rešerši kontaktních přístupových systémů a porovnáním možností identifikace technologiemi s kontaktním prvkem.

Dále se práce zabývá návrhem přístupového systému s využitím kontaktních identifikačních čipů iButton.

Praktická část byla realizována v podobě funkčního prototypu přístupového systému. Součástí práce je rovněž realizace obslužné počítačové aplikace ve SCADA prostředí Promotic. Realizovaný systém bude využit pro potřeby výuky v laboratořích Katedry telekomunikační techniky na Fakultě elektrotechniky a informatiky VŠB-TU Ostrava.

Klíčová slova

Přístupový systém; kontaktní čtečka; čip Maxim iButton; řízení přístupu, mikrokontroler; datová sběrnice; 1-wire; RS485; signál; Promotic

Abstract

The diploma thesis deals with the design and implementation of the access system with 4 Maxim iButton chips readers.

The aim of this work was to create a contact control system based on Microchip PIC microcontrollers. The theoretical part is devoted to the research of contact access control systems and the comparison possibilities of identification technologies with the contact element.

It also deals with the design of an implementation system using iButton contact identification chips.

The practical part was realized as a working prototype of the access control system. Part of the thesis is also the implementation of service computer application in the Promotic SCADA environment. The created system will be used for teaching purposes in the laboratories of the Department of Telecommunications in the Faculty of Electrical Engineering and Computer Science at VŠB-TU Ostrava.

Key words

Access control system; contact reader; Maxim iButton chip; access control, microcontroller; data bus; 1-wire; RS485; signal; Promotic

Seznam použitých symbolů a zkratek

Symbol	Jednotky	Význam symbolu
U	V	Napětí
T	s	Čas
R	Ω	Odpor
f	Hz	Frekvence
Um	mV	Amplituda
C	F	Farad

Zkratka	Význam
WBskript	Skriptovací jazyk
C	Programovací jazyk
IO	Integrovaný obvod
DIP	Dual in-line package, pouzdro pro IO s roztečí vývodů 2,54 mm
PDIP	Small Outline Integrated Circuit
SOIC	Small Outline Integrated Circuit, pouzdro pro IO s roztečí vývodů 1,27 m
TSOP	Thin small outline package , pouzdro pro IO s roztečí vývodů 0,55 mm
LED	Light-Emitting Diode, dioda emitující světlo
OLED	Organic LED, displej s organických LED
ROM	Read-Only Memory, typ počítačové paměti
RAM	Random Access *Memory, typ počítačové paměti*
EEPROM	Electrically Erasable Programmable ROM, *typ počítačové paměti*
RFID	Radio Frequency Identification, identifikační technologie
USB	Universal Serial Bus, typ seriové sběrnice
UART	Universal Asynchronous Receiver and Transmitter, typ seriového rozhraní
SPI	Serial Peripheral Interface, typ seriového rozhraní
I2C	Inter-Integrated Circui, typ seriového rozhraní
TTL	transistor-transistor-logic, snandard pro implementaci digitálních IO
XLP	eXtreme low power, technologie mikrokontrolerů s nízkou spotřebou

Obsah

1 KONTAKTNÍ PŘÍSTUPOVÉ SYSTÉMY

Úvodem je nutné vymezit samotný pojem přístupový systém. Přístupovým systémem, někdy označovaným také jako systém kontroly vstupů, se obvykle v odborné literatuře rozumí určitý soubor opatření, jehož cílem je zajistit řízení a evidenci přístupů buď do zabezpečeného objektu nebo do konkrétních prostor, a to na základě jednoznačně definovaných a jednotlivým přistupujícím osobám přidělených práv. Jinak řečeno, přístupový systém umožňuje těm osobám, které mají příslušné oprávnění vstup do určitých prostor, popř. i jejich opuštění, zatímco osobám bez potřebného oprávnění je prostřednictvím přístupového systému ve vstupu do určitých prostor zamezeno. [1]

Přístupové systémy mohou mít řadu podob, kterými jsou opatření systémová, mechanická, fyzická a elektronická. Předkládaná diplomová práce se primárně zaměřuje na elektronické kontaktní přístupové systémy se specializací zejména na systém, který využívá čtečky Maxim iButton.

1.1 Vymezení pojmu elektronický přístupový systém

Elektronický přístupový systém je možné chápat jako podmnožinu informačního systému, která zajišťuje, aby do určitých prostor měl přístup jen ten, kdo má potřebná oprávnění. Jde tedy vlastně o počítačem řízený soubor prvků, který kontroluje přístupy do vymezeného prostoru. Elektronický informační systém si lze představit jako *„elektronickou podobu klíčů od dveří.“* Elektronický systém je však ve srovnání s klíčem od dveří výrazně efektivnější. V zásadě lze shrnout, že elektronický informační systém sleduje následující tři aspekty: kdo se kdy a kam v určitém objektu dostane. [1]

Velkou výhodou elektronického přístupového systému ve srovnání s jinými než elektronickými přístupovými systémy je pak skutečnost, že pokud např. dojde ke ztrátě přístupové karty, lze snadno změnit naprogramování tak, aby se případný cizí člověk do prostor nedostal. Stejně tak ve firmách je možné snadno monitorovat pohyb zaměstnanců, ale třeba i návštěv, které do budovy vstupují. Lze tak snadno získat přehledy z databáze systému, které poskytují nejen údaj o tom, kdo, kdy a kam šel, ale také například o docházce zaměstnanců. [2]

1.2 Stručná historie elektronických přístupových systémů

Jak již plyne z předcházející kapitoly, přístupové systémy mechanické či fyzické jsou záležitostí, která se používala od pradávna. Úplně první přístupové systémy jsou zdokumentovány ve starověkém Egyptě. Šlo o dřevěné závory, jež byly za účelem zamezení přístupu neoprávněných osob opatřené systémem západek. Tehdejší alternativu klíče představovaly destičky s kolíky, které se vsouvaly do dřevěné závory, čímž docházelo k nadzvedávání dřevěné závory. Nebyl to však jen Egypt, prakticky všechny civilizace v historii obdobu tohoto systému v různých modifikacích využívaly.

Na systém západek pak navázal o něco složitější mechanismus klíče a zámku, které jsou vzájemně kompatibilní pouze tehdy, pokud se vloží správný klíč do správného zámku. Klíče a zámky jsou využívány dodnes, v průběhu vývoje však prošly zásadním vývojem a modernizací. I tak se ovšem

v moderní době ukázalo, že klíče lze ztratit, čímž nejsou zdaleka tak spolehlivým přístupovým systémem jako elektronický přístupový systém. [1]

Elektronické přístupové systémy jsou však neoddělitelně spjaty s moderní dobou a s rozvojem informačních technologií. Praxe instalace elektronických přístupových systémů tak započala v 60. letech 20. století. Prvotním cílem bylo právě odstranit velmi časté problémy se ztracenými klíči. První elektronické přístupové systémy byly ve srovnání s aktuálními velmi jednoduché a fungovaly na principu jednoduché klávesnice a PIN kódu, který osoba s oprávněním vstupu do daných prostor dostala a za účelem vstupu zadávala na klávesnici. Ačkoliv se jedná o systémy velmi jednoduché, stoj za zmínku, že i v dnešní době je někteří lidé i podniky stále ještě používají. Další možností na přelomu 70. a 80. let bylo využití tzv. přístupových karet. Přístupové karty se používaly společně se čtečkou karet pro řízení přístupu. I tyto systémy byly ve svých počátcích velmi jednoduché a nebyly schopné například rozlišovat jednotlivce s různým typem oprávnění.

K zásadnímu rozvoji elektronických přístupových systémů došlo pak v souvislosti s rozvojem ostatních počítačových technologií zejména na počátku 70. let 20. století. Rozvoj elektronických přístupových systémů vedl k tomu, že na konci 80. let se začaly využívat v dnešní době běžné bezkontaktní přístupové karty a radiofrekvenční systém identifikace (RFID technologie), který umožňoval identifikaci osob pomocí radiofrekvenčních technologií [4].

Na přelomu 80. a 90. let 20. století pak začal být kladen důraz na rozlišování nejrůznějších rolí prostřednictvím elektronických přístupových systémů. To znamená, že systém identifikuje při přístupu (např. na základě přístupové karty, kde je nahrán určitý údaj) různé role. To znamená, že například generální ředitel podniku má přístup do všech prostor v podniku, zatímco skladník například jen do skladů a zázemí k nim příslušejícího. Právě na přelomu 80. a 90. let začal být na tuto možnost diferenciace rolí kladen velký důraz. Role byla tehdy spjata právě s postavením či pracovní pozicí v organizaci, což bylo spojeno s faktem, že systémy založené na odlišování rolí byly ještě na počátku 90. let 20. století velmi jednoduché.[6]

V roce 90. let pak došlo k dalšímu významnému rozvoji, který byl spjat s velkým rozvojem a rozšířením internetu a WWW (world wibe webu), kdy bylo možné přístupové systémy propojit právě s internetem. Spojení přístupových systémů s internetem také umožnilo rozvoj dalších funkcí a služeb, které elektronické kontaktní systémy nabízejí.[5]

V průběhu posledních dvaceti let došlo k zásadnímu rozvoji a rozšíření možností funkcí elektronických přístupových systémů stejně jako k rozvoji jejich bezpečnosti a spolehlivosti. Přístupové karty či bezkontaktní přístupové karty se v posledních letech používají ve spojení s čtečkou karet pro řízení přístupu. Taková čtečka karet je připojena k inteligentnímu ovladači dveří, který obsahuje uložené informace o programování ze softwaru řízení přístupu, o tom, komu je vstup do určitého prostoru povolen a v jakém časovém úseku. Systém dnes obvykle má celou řadu funkcí, mezi které patří typicky například sledování docházky zaměstnanců. [4]

Z hlediska vývoje přístupových systémů je nutné zmínit, že obecně platí, že došlo k zásadnímu snížení cen přístupových systémů, čímž jsou snadno dostupné i pro menší firmy, popř. i pro domácnosti. Zároveň systémy nabízejí výrazně více funkcí a plní tak více úkolů. [3]

1.3 Rozvoj systémů založených na technologii kontaktních čteček

V současné době existuje celá řada způsobů, kterými se může jedinec identifikovat v rámci elektronického přístupového systému. Za zmínku stojí především manuální způsoby identifikace (k identifikaci se využívá např. vypínač nebo kódový zámek), dále existují systémy magnetické (k identifikaci se používá karta s magnetickým proužkem, ve kterém jsou nahrané potřebné informace). Další možností jsou pak optické formy identifikace (zde se používá k identifikaci buď čárový kód nebo QR kód případně matrix 2 D nebo kruhový kód) nebo biometrické způsobe identifikace (založené např. na otisku prstu, oční duhovky či sítnice). Poslední běžnou možností identifikace je identifikace prostřednictvím čipu, který je uložen v nějakém integrovaném obvodu, ve kterém je možné provádět jak čtení dat, tak je možné do něj data zapisovat. [4]

1.3.1 Čipová identifikace

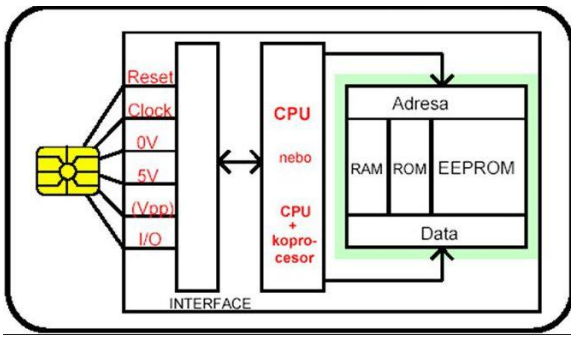
Čipová forma identifikace je nejčastěji založena na tom, že jedinec má k dispozici nějaké zařízení jako např. čipovou kartu, která se podobá například běžné platební kartě nebo jiné médium – např. čip zabudovaný v kovovém obalu (jako je tomu např. u čipu iButton).

Identifikaci za pomocí čipu může probíhat buď kontaktně nebo bezkontaktně. V případě kontaktní formy identifikace je nezbytný přímý kontakt média obsahujícího čip se čtečkou (tj. například jeho přiložení na čtečku). K přenosu informace zde totiž dochází prostřednictvím vodících kontaktů čtečky a karty či jiného média, jehož součástí je například čip. iButton, který bude představen níže je typickým představitelem čipu, který se používá při kontaktní formě identifikace. Druhou možností je pak identifikace bezkontaktní. Zde jsou data na čipu směrem ke čtečce přenášena bezdrátově, k čemuž se využívá elektromagnetické pole, které čtečka vyřazuje, není tudíž nutné čipovou kartou či jiný nosič nikam přikládat, postačuje, pokud se jedinec, který má kartu u sebe dostane do oblasti elektromagnetického pole čtečky. Vedle toho existují také kombinované systémy. [5]

1.3.1.1 Čipová karta

Kontaktní čipová karta se vyznačuje tím, že je vybavena celkem osmi kontaktními body, které jsou umístěny na kontaktních ploškách. Každý kontaktní bod má svůj účel. Některé se používají pro napájení, jiné pro sériovou komunikaci, další pro přívod externího taktovacího signálu případně pro přívod rogramovacího napětí.
U kontaktní karty je vždy důležité, aby měla především kontakty k propojení se čtečkou.[6]

Obrázek č. 1: Architektura čipové karty



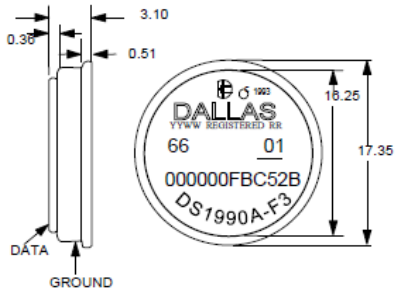
Zdroj: [15]

Architektura čipové karty je přehledně znázorněna v obrázku č. 1.

Zde je nutné poukázat také na rozvoj, kterým kontaktní čipové karty od svých počátků prošly. S prvními kontaktními čipovými kartami se totiž bylo možné setkat v Německu na konci 60. let, kdy Helmut Gröttrup a Jürgen Dethloff vytvořili vůbec první čipovou kartu, na kterou pak za spolupráce s Dethloff navázala společnost Honeywell a první čipovou kartu s mikroprocesorem, která tak vznikla, si následně v roce 1978 nechal Dethloff patentovat. K nejvýznamnějšímu rozvoji v oblasti kontaktních čipových karet pak došlo v 90. letech 20. století, kdy velmi výrazně narůstala i možnost jejich praktického využití. Zatímco dříve byly kontaktní čipové karty jednoduché, byly osazené pouze paměťovými registry, postupně byla přidávána nejrůznější rozšíření, což vedlo k tomu, že dnes se lze setkat i s kontaktními čipovými kartami, které obsahují přímo v čipu například USB rozhraní. [5]

1.3.1.2 Čip iButton jako příklad jiného média a čtečky

Přístupové čipy iButton (nebo také Maxim's iButton, popř. Dallas čipy) patří v současnosti v českém prostředí k nejpoužívanějším čipům v přístupových systémech, byť je nutnost kontaktu s rozhraním určitým limitem, které vede k tomu, že se tyto čipy postupně nahrazují bezkontaktními alternativami. Vyvinula je společnost Dallas Semiconductur v USA, kterou následně koupila společnost Maxim, proto je používáno hned několik názvů, pod kterými se rozumí čipy iButton. Jednoduše řečeno lze říci, že iButton je vlastně počítačovým čipem, který je uzavřený v ocelovém obalu o průměru 16 mm, který je možné umístit takřka kamkoliv, což je dáno jednak miniaturní velikostí čipu, ale také jeho vysokou odolností jak vůči mechanickému namáhání, tak i vůči elektromag. poli. Charakteristickým rysem iButton je, že je to právě jeho pouzdro, které používá jako elektronické komunikační rozhraní (podobné jako kontaktní body umístěné na čipové kartě). [7]
Pouzdro čipu DS1990 je na obrázku č. 2.



Obr. č.2 pouzdro původního čipu Dallas iButton DS1990

Zdroj: [16]

V současnosti se využívá okolo 20 různých druhů IButtonů, přičemž každý z nich má k dispozici trochu jiné doplňkové funkce. Základní členění iButtonů je následující: [8]

- „Address Only (základní verze, obsahuje 64 bitů ROM)
- Memory (NV RAM, EPROM, EEPROM)
- Real-Time Clock (kalendáře, hodiny, stopky apod.)
- Secure (bezpečnostní zařízení jako např. elektronické zámky, parkovací hodiny, softwarová autorizace a další)
- Data Logger (záznam teploty, vlhkosti apod.)“

Jak je zřejmé z výše uvedeného rozdělení mezi základním čipem, který stál na samotném počátku a současnými čipy, které zvládají celou řadu úkolů je významný rozdíl. Pro účely přístupových systémů je nejvhodnější verze Secure. Navzdory skutečnosti, že vývoj, kterým iButton prošel je velmi zásadní, stojí za zmínku, že iButton stejně jako sběrnice a čtečky dat s ním kompatibilní (tj. využívající komunikační rozhraní 1-Wire) vznikl až v 90. letech 20. století [8].

1.3.2 Rozvoj čteček

Čipy iButton využívají za účelem komunikace proprietární komunikační rozhraní 1-Wire. Jedná se o sériový protokol, který při komunikaci využívá jeden pár vodičů, konkrétně se jedná o datový vodič a dále o společnou zem. To znamená, že vedle přenosu dat jsou zařízení také napájena. Komunikace má podobu Master-Slave, u které je Master iniciátorem a zároveň řídící jednotnou komunikace s podřízenými stanicemi. Zařízení je vždy také vybaveno jedinečným 64 bitovým číslem, které slouží k identifikaci a podle konkrétního provedení může být vybaveno rovněž pamětí, popř. koprocesorem. Komunikace přitom probíhá obousměrně, asynchronně a je také poloduplexní. [7]

Stejně jako čipů iButton je i čteček, které jsou s těmito čipy kompatibilní celá řada a je možné volit mezi zcela jednoduchými čtečkami, ale i mezi čtečkami, které nabízejí velké množství doplňkových funkcí.

Jako první je potřeba představit starší typy čteček, mezi které patří na českém trhu dostupná čtečka DSRS2130, která je na obrázku č. 3, a čtečky DSRS2319 a DSRS2333. Tyto čtečky byly postaveny na přenosu 48-bitového kódu směrem do počítače prostřednictvím sériové linky RS232 nebo přes USB. Jednalo se vesměs o zařízení sestavená z kontaktní čtecí plochy, která byla upevněná na plastové skříňce, jejíž rozměry činily 48 x 40 x 22 mm. Hned vedle čtecího kontaktu se nacházela je indikační dioda LED. [9]

Tyto čtečky byly z dnešního pohledu velmi jednoduché a v současné době se již takřka nepoužívají. V současné době se lze setkat buď se čtečkou DSRS2402 s výstupním relé nebo se čtečkami řady 2400. První z uvedených čteček, která nese označení DSRS2402 je elektronické zařízení, které je sestavené z kontaktní čtecí plochy, která je shodně se staršími typy čteček upevněná na plastové skříňce společně s indikační diodou LED. Vedle toho však tato čtečka sestává ještě z druhé skříňky, v níž je umístěna řídicí elektronika, jenž obsahuje výstupní relé společně s přepínacím kontaktem. Tento typ čtečky je vybavený jednočipovým mikroprocesorem, jehož úloha spočívá v tom, že zajišťuje obsluhu čtení kódu z čipu a rovněž jeho porovnání s kódy čipů, které jsou uloženy v interní paměti tohoto zařízení. V případě, že je kód čteného čipu totožný s některým z kódů, které má zařízení uložené ve své paměti, dojde na určitou dobu

Obrázek č. 3: Čip iButton a čtečka DSR2319

Zdroj: [9]

k sepnutí výstupního relé. Tento typ čtečky je napájen stejnosměrným napětím 24 V. [9]

Nejnovějšími a tím i nejmodernějšími čtečkami jsou pak čtečky řady 2400. Dostupné jsou tři typy čteček *(*DSRS2430, DSRS2419, DSRS2433 a DSRS2433L*)* jejichž jednotný přenosový protokol je ve formátu ASCII a lze je snadno implementovat do jiných systémů. Jsou také vybaveny komunikačním rozhraním RS232, které umožňuje připojení čtečky k počítači.

Mimo přenos sériového čísla čipu lze ovládat LED diodu, zablokovat činnost čtečky a rovněž přečíst typové číslo čtečky. Volně dostupný ovládací software umožňuje jak čtení kódu čipů do vlastního okna, tak i do jiného aktivního okna na počítači. Na zakázku je možné upravit software podle přání uživatele. Na objednávku lze dodat čtečky i v jiném provedení (bez skříňky, čtecí plocha připojitelná přes konektor, ...). Rovněž lze vyrobit čtečky s rozhraním RS485 a mít tak na jedné komunikační lince až 128 čteček“ [9].

Moderní čtečky se vyznačují tím, že jsou dodávány v podobě kompletního zařízení, a to v plastové skříňce s pevně připojeným komunikačním kabelem. Tento kabel je standardně dlouhý 1,5 m. Na zakázku je pak také možné dodat i odlišnou délku kabelu, a to v délce od 0,5 do 3 m. Lze však čtečku dodat i s kabelem výrazně delším, kdy je čtečka již vybavena i komunikačním rozhraním s proudovou smyčkou. [9]

1.4 Uplatnění technologie využívající přístupové čipy iButton v současnosti

Jak již bylo nastíněno, technologie iButton je na jednu stranu poměrně oblíbená a běžně využívaná, na druhou stranu je její nevýhodou, že se jedná o technologii kontaktní, která je v dnešní době vytlačována bezkontaktními technologiemi, což se v přístupových systémech projevuje velmi výrazně.

Kontaktní přístupové systémy jsou dnes obvykle vystavěny na terminálech, které leckdy kombinují jak čtečku pro iButton čipy, tak i čtečku pro další velmi oblíbenou a prověřenou technologií čipů, kterou jsou RFID čipy. Velmi často je využívána rovněž to, že moderní terminály umí fungovat jak online, tak off-line a komunikuje průběžně s PC, čímž není určen jen k umožnění přístupu do určité oblasti, ale leckdy také třeba sleduje přítomnost pracovníků na pracovišti, jejich pohyb v rámci firmy a celou řadu aspektů, které si zaměstnavatel následně má možnost stáhnout do počítače a analyzovat a vyhodnocovat. Terminály dnes standardně mají rovněž interní paměť, která umožňuje data stáhnout a analyzovat i do určité doby (např. po dobu jednoho měsíce) zpětně. [10]

Samozřejmostí je pak jednak to, že terminály umožňují otevírání dveří, závor nebo třeba ovládání tlačítek ve výtazích v určité budově či firmě, ale je možné využít také funkci, která dává možnost snadno nastavit nejrůznější jednotlivá omezení týkající se identifikačních čipů s ohledem na konkrétní dny či časové rozmezí (tj. např. terminál umožní přístup na pracoviště v pondělí – pátek mezi 6:00 - 24:00, ale v ostatních časech a dnech v týdnu nikoliv). Řada terminálů dnes umožňuje rovněž zvukovou signalizaci.

Příkladem takového terminálu, který je kompatibilní s čipy iButton je například Autonomní přístupový terminál CX303 či CX105. [10

2 Rešerše kontaktních přístupových systémů

V elektronických kontaktních systémech se nejčastěji využívají čtyři různé technologie, kterými jsou čip iButton, přístupové karty s magnetickým proužkem, přístupové karty s čipem a karty s čárovým kódem. V této práci je pojednáno pouze o kontaktních verzích přístupových karet, byť je nutné zmínit, že zatímco iButton existuje pouze v kontaktní variantě, přístupové karty mohou existovat jak ve variantě kontaktní, tak ve variantě bezkontaktní, nicméně v následujícím textu jsou sledovány pouze karty kontaktní.

2.1 Srovnání čipu iButton s kontaktními kartami

Prvním rozdílem, kterým se čip iButton liší od technologií, které využívají kontaktní karty, je vlastní identifikační médium, tedy fakt, že čip iButton je uzavřen v ocelové plechové schráně, která je pak obvykle ukotvena v plastovém přívěšku, zatímco u kontaktních karet je čip či magnetický pásek, případně čárový kód, na němž jsou uloženy požadované informace obsažen v plastové kartě (obvykle z PVC).

Všechny tři varianty kontaktních karet jsou vystavěny na stejném principu, tj. nosič informací (nezávisle na tom, zda se jedná o čip, magnetický pásek nebo čárový kód) je vložen do plastové karty, zatímco iButton je vzhledově odlišný a vystavěný na jiném principu, kterým je čip uvnitř ocelového pouzdra MICRO CAN.

S výše uvedeným pak souvisí například i vyšší odolnost vůči mechanickému namáhání stejně jako delší životnost čipu iButton ve srovnání se všemi třemi uvedenými technologiemi kontaktních karet.

Ve srovnání s kontaktními kartami bývají čipy iButton velmi oblíbené zejména proto, že jsou odolné, ale vyznačují se rovněž vyšší mírou spolehlivosti ve srovnání s kartami. Zejména v náročnějších provozech jsou totiž karty velmi intenzivně namáhány a není neobvyklé, že dochází k jejich zničení či poškození, což se u čipů iButton takřka neděje. I v případě, že je s čipem zacházeno velmi nešetrným způsobem, čip iButton se v naprosté většině případů nepodaří poškodit, zatímco kontaktní karty není problém například mechanickým namáháním (ohybem) poškodit natolik, že zcela přestanou fungovat. [12]

S ohledem na skutečnost, že čipy iButton vykazují jen minimální míru opotřebení, je také možné je přidělovat oproti záloze, což by v případě karet, které se vyznačují významně větší mírou opotřebení mohlo vést spíše ke konfliktům v podniku.

Rozdíl je také ve vlastní manipulaci s čipem a kontaktními kartami, neboť čip iButton se vyznačuje velmi jednoduchou manipulací, protože na rozdíl od karet není u čipu iButton nutné přikládat čip k zařízení příliš precizně, stačí jen krátký dotek, u kterého nezáleží na tom, jak je čip otočen, zatímco kartu je do čtečky nutné přikládat na přesné místo, dotyk tak trvá déle a v některých případech je dokonce nutné přizpůsobit rychlost přiložení čtečce, jinak čtečka kartu nedokáže přečíst. [12].

2.2 Srovnání čipu iButton s jednotlivými druhy karet

Vedle všeobecných charakteristik, které odlišují čip iButton od kontaktních karet obecně, je pak nutné přihlížet i k dílčím rozdílům, které existují mezi jednotlivými kontaktními kartami navzájem. V řadě odlišností se však projevuje i obecný rozdíl mezi čipem iButton a kontaktní kartou, neboť například v případě ceny, je zřejmé, že zatímco iButton stojí okolo 20 Kč/kus, v případě karet se cena pohybuje od 10 do 200 Kč. Za cenu 10 Kč je dostupná nejjednodušší karta s čárovým kódem, 30 Kč pak stojí karta magnetická, která je v dnešní době již leckdy vnímána jako zastaralé řešení a 35 - 200 Kč pak karta čipová, která je patrně nejoblíbenější a výše ceny závisí i na míře zabezpečení.

Za důležité je možné považovat i to, že nejstarší technologií v oblasti kontaktních přístupových systémů je karta s magnetickým proužkem, na kterou pak rychle navázala karta s čipem, které do dnes patří k nejoblíbenějším. V současné době se ve srovnání s historií více používají zejména lépe a více zabezpečené karty s čipem, dříve byly karty jednodušší, o něco později, a to v 70. letech se objevily karty s čárovým kódem, čip iButton začal být vyráběn a byl patentován až v 90. letech, čímž je jednoznačně nejnovější a posledně s moderními typy dražších čipových karet také nejmodernější technologií, neboť shodně s čipovými kartami umožňuje vysokou míru zabezpečení i přidání celé řady různých funkcí nad rámec základní funkce.[13]

Významný rozdíl je také v délce bitového klíče, iButton společně s kartou s čárovým kódem nabízí délku bitového klíče 64 bitů a karty s čipem obvykle v dnešní době nabízejí 128 bitů. [7]

Použití všech čtyř zvolených kontaktních přístupových systémů je velmi podobné. Je možné je použít jak za účelem přístupu (a s tím související identifikace osoby), tak za účelem zamezení přístupu do určitých prostor, ale je možné je využívat i za účelem evidence docházky a pracovní doby. Stejně tak je možné však všechny technologie využít i jinak, například jako elektronické peněženky, abonentky a věrnostní karty. Nelze zde tedy identifikovat zásadní rozdíl mezi jednotlivými variantami. [14]

Významné odlišnosti jsou však v jednotlivých výhodách a nevýhodách, které tyto technologie nabízejí. U těch nejjednodušších je výhodou prakticky jen nízká cena a snadná identifikace. To platí například po karty s magnetickým páskem či čárovým kódem. Oproti tomu iButton nabízí výrazně více výhod, kterými je jednak již výše zmíněná odolnost a snadné používání, kdy není potřeba jej přikládat nějak přesně, ale třeba i velké množství doplňkových funkcí, které lze na čipy nahrát. V souvislosti s odolností a nezničitelností je pak s i vysoká míra spolehlivosti uložených dat. V případě čipových karet je vekou výhodou zejména možnost asymetrického kódování, podobně jako u iButtonu je možné zde přidávat celou řadu dalších prvků, které zajišťují jak bezpečnost, tak i doplňkové funkce.

Při vlastním výběru zvolené varianty je ovšem vždy třeba sledovat i nevýhody. Největší nevýhody byly zaznamenány u karet s magnetickým proužkem, které jsou poměrně snadno padělatelné, data se navíc dají snadno zkopírovat a s ohledem na to, že kartu lze snadno zničit např. v magnetickém poli, je zde i problém se zachováním dat. Podobného výsledku dosahovaly i karty s čárovým kódem. Lépe na tom pak jsou karty čipové, nicméně zde je nutné rozlišovat dražší varianty, které mohou být poměrně zatěžují a levné varianty, které jsou méně bezpečné. Problémem je i omezená životnost čipového kontaktu. Oproti tomu u iButtonu se jako nevýhoda projevil pouze vyšší náklad a v některých prostředích možnost oxidace, což ovšem nebude většinový problém. [7] [13] [14]

Nejdůležitější údaje pro srovnání jsou zaneseny rovněž do přehledné tabulky č. 1 této práce.

Tabulka č. 1: Srovnání základních charakteristik čipu iButton a kontaktních karet

	iButton	Karta s magnetickým proužkem	Karta s čipem	Karta s čárovým kódem
Kapacita media /délka ID klíče	-/64 bits	160Bytes/- (read only)	16KB /128 bits	32 bits (*Interleaved* 2/5) 1 KB
Patentováno	1992	1968	1968	1970
Číslo patentu	US6016255A	US4253017A US3685690A	DE 2560689C2	US2612994(Int 2/5) US5243655(PDF 417)
Vynálezce	Michael Bolan, Nicholas M.G. Fekete	Marion R Karecki George R Chastain Thomas R Barnes (USA)	Roland Moreno (FR)	David Allais (Interleaved 2of5) WANG YNJIUN (Code PDF 414)
Současný vlastník patentu	Maxim Integrated Products Inc	Docutel Corp	INTERNATIONALE POUR L'INNOVATION PARIS FR Ste	Symbol Technologies LLC (Code PDF 414)
Nasazeno na trh	90. léta	1970	1969	1973
Cena média	20 Kč	20 Kč	35 - 200 Kč	10 Kč
Příklady použití	Docházkové systémy Identifikace Přístupové systémy Elektronická peněženka	Přístupová karta Identifikace Evidence pracovní doby Banky (na ústupu) Věrnostní programy	Přístupová karta Identifikace Banky Sim karty Telefonní karty Abonentky	Identifikace předmětů Identifikace osob Přístupové systémy Elektronická peněženky
Výhody	Odolný proti poškození Spolehlivost uložených dat Snadné používání Doplňkové funkce	Nízká cena Snadná identifikace	Spolehlivost Bezpečnost dat Asymetrické kódování Možnost různě kombinovat s dalšími prvky	Nízká cena Snadné identifikace
Nevýhody	Vyšší náklad Možnost oxidace	Snadná zničitelnost Možnost poškození magnetického pruhu Snadné kopírování	Snadná zničitelnost Omezená životnost kontaktu Vyšší cena při lepším zabezpečení	Snadná padělatelnou Nižší míra odolnosti Snadné poškození Nižší míra bezpečnosti

Zdroj: Vlastní zpracování dle [14]

3. Návrh koncepce přístupového systému

Vlastní návrh přístupového systému vychází z poměrně úzce specifikovaného zadání a zohledňuje požadavky kladené jednak na to, aby zařízení sloužilo k demonstraci možností využití mikrokontrolerů a souvisejících HW a SW prostředků pro potřeby výuky, dále pak zohledňuje některé požadavky související s nároky na reálné provozní nasazení přístupového systému.

3.1 Koncepce systému

Systém má disponovat 4 čtečkami a má být navržen tak, aby fungoval autonomně, bez komunikace s dalšími zařízeními výpočetní techniky. Dále má být umožněn dispečerský dohled nad fungováním systému prostřednictvím osobního počítače. K dohledu nad fungováním přístupového systému a administraci uživatelů má sloužit obslužná aplikace realizovaná v prostředí Promotic. Obslužná počítačová aplikace bude rovněž zaznamenávat historii přístupů do jednotlivých místností a umožní individuální nastavování přístupových práv uživatelům, resp. iButton čipům pro jednotlivé vstupy do místností.

Návrh přístupového systému bude splňovat kritéria kladená na kontaktní přístupový systém vhodný pro řízení a monitoring přístupů do jednotlivých místností rozsáhlejšího objektu. Realizovaný přístupový systém pak bude zhotoven ve formě funkčního prototypu a bude sloužit pro potřeby výuky na Katedře telekomunikační techniky FEI VŠB-TU Ostrava.

3.2 Požadavky na přístupový systém

Níže uvedeme výčet požadavků, které bude navržený systém po realizaci a implementaci všech částí, splňovat. Relizovaný přístupový systém bude tedy:

- **monitorovat požadavky na vstupy do jednotlivých místností – přečíst a zobrazit ID přiložených iButton čipů**
- **evidovat povolené iButton čipy, přiřazovat tyto čipy uživatelům a nastavovat jim přístupová práva**
- **vyhodnocovat požadavky na vstup do jednotlivých místností, signalizovat povolení vstupu a zamítnutí vstupu signalizačními LED u každé ze čteček**
- **zobrazovat aktuální dění na čtečkách v dohledové aplikaci v počítači**
- **prostřednictvím počítačové aplikace měnit nastavení přístupových práv**
- **v počítači ukládat informace o historie přístupů do jednotlivých místností**

- **evidovat historii přístupových interakcí jednotlivých čipů**
- **evidovat zamítnuté požadavky na vstup**
- **vyvolat alarmní událost při pokusu o přihlášení neevidovaným čipem**
- **umožnit krátkodobý provoz systému ze záložního zdroje pro případ výpadku napájení ze sítě**
- **zabezpečit systém před následky pokusů o zničení vnějším narušitelem – zabezpečit kontaktní pole před poškozením zkratem a před zničením při připojení zdroji elektrického proudu na kontakty snímacího pole čtečky**

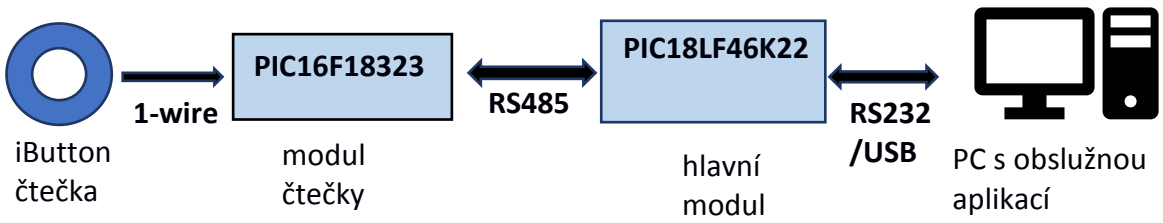
3.3 Požadavky na HW - mikrokontrolery

V návrhu systému nejdříve definujeme požadavky na hardware, neboť vývoj softwaru je možný až po výběru HW jednotek a komunikačních sběrnic.

Na realizaci jednotky čtečky využijeme mikrokontrolery disponující rozhraním UART, které by dále měly splňovat kritéria jako je nízká spotřeba eklektické energie a nízká pořizovací cena.

Pro vytvoření řídící jednotku budeme potřebovat mikrokontroler který má dvě rozhraní UART, jedno pro komunikaci se čtečkami a druhé pro komunikaci s počítačem.
Po konzultaci s vedoucím diplomové práce byly pro konstrukci logické části přístupového zařízení zvoleny mikrokontrolery PIC od firmy *Microchip* Technology, konkrétně vývojová DM164134 osazena procesorem PIC18LF46K22, dále pak mikrokontrolery PIC18F26K22 a PIC16F18323.

Ke zhotovení funkčního vzoru kontaktního přístupového systému budou tedy využity procesory PIC16F18323 jako moduly čteček a vývojová deska DM164134 s PIC18F46K22 jako Master



Obrázek č.4 Blokovové schéma HW realizace přístupového systému
Zdroj: vlastní

jednotka. Mikrokontrolery pro moduly čteček a další komponenty budou osazeny do nepájivého pole.

3. 4 Požadavky na hardware - komunikační sběrnice

Z předchozí definice požadovaných vlastností přístupového systému vyplívají nároky nejen na samotné mikrokontrolery, ale především na řešení komunikačních sběrnic.

Pro samotné přečtení adresy iButton čipu poslouží sběrnice 1-wire která byla za tímto účelem vyvinuta původním výrobcem identifikačních čipů iButton, firmou Dallas Semiconductor.

K realizaci sběrnice sloužící k následného odeslání přečtené adresy přiloženého iButton čipu z modulu čtečky do hlavního modulu již je třeba vzít v úvahu konkrétní výše uvedené požadavky. Je třeba zabezpečit to, aby moduly čteček byly schopny komunikovat s řídícím modulem i v rámci rozlehlejších budov /tedy na vzdálenosti v jednotkách stovek metrů/ a zamezit možnému nežádoucím zarušení komunikace od vnějších zdrojů rušení. Pro komunikaci mezi řídícím modulem a moduly čteček bude vhodné použít sběrnici, která výše uvedené nároky splňuje. Jako velmi vhodné se jeví využití sběrnice RS 485. Pro realizaci sběrnice RS 485 bude třeba použit UART/RS485 převodníky, k tomuto účelu lze využít integrovaný obvod SN75176.

Ke komunikaci hlavního modulu a osobního počítače bude třeba použit převodník z rozhraní UART mikrokontroleru na sériové rozhraní osobního počítače, tedy RS232. K tomuto účelu bude použit 4-pinový kabel s integrovaným převodníkem UART/USB.

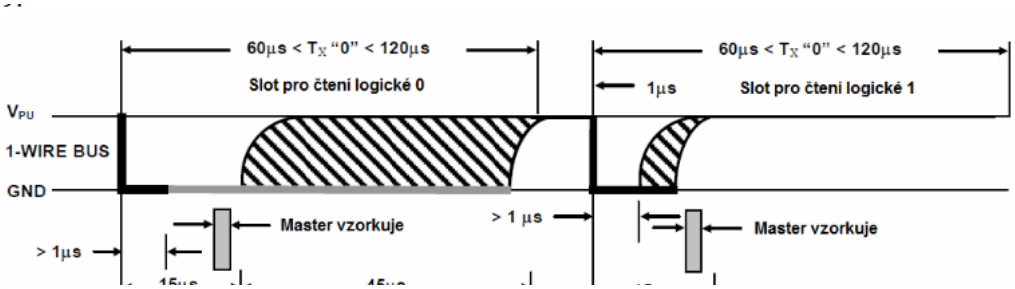
3.4.1 Sběrnice 1-WIRE

Sběrnice 1-Wire umožňuje připojit k mikrokontroleru několik zařízení prostřednictvím pouhého jednoho vodiče, tato sběrnice byla navržena firmou Dallas Semiconductor.

Sběrnice má vždy jeden řídící obvod (master) a jeden nebo více podřízených obvodů (slave). Všechny obvody jsou připojeny na společnou zem a také paralelně na společný datový vodič, který bývá připojen přes rezistor 4,7 kΩ na napájecí napětí, a tím nastaví sběrnici na vysokou úroveň. Data jsou na sběrnici vysílána v tzv. „time slotech“. Jeden slot je dlouhý 60 až 120 µs a během jednoho slotu je vysílán nebo přijat jeden bit. Mezi jednotlivými sloty musí být mezera minimálně 1 µs. Sloty lze rozdělit na čtyři druhy a to zápis 1, zápis 0, čtení 1 a čtení 0. Komunikace na sběrnici je realizována jako asynchronní a poloduplexní. Komunikaci zahajuje vždy master reset pulsem. [y1]

Zahájení komunikace na sběrnici 1-Wire reset pulsem (převzato z [22]). Master nejprve nastaví datový vodič na nízkou úroveň a čeká 480 µs, poté sběrnici uvolní a čeká 70 µs. Odpor vrátí sběrnici zpět na vysokou úroveň. Pokud je na sběrnici připojeno nějaké zařízení, tak detekuje vzestupnou hranu a po prodlevě 15 až 60 µs nastaví datový vodič na 60 až 240 µs na nízkou úroveň. Jestliže se zařízení

správně ohlásí, může master začít vysílat a přijímat data. Při zápisu logické nuly master stáhne datový vodič na nízkou úroveň a čeká 60 µs, poté sběrnici uvolní a čeká 10 µs. Při zápisu logické jedničky master stáhne datový vodič na nízkou úroveň a čeká 6 µs, poté sběrnici uvolní a čeká 64 µs. Průběh pro čtení dat je uveden na obr.5.



Obr. č. 5 : Přijem dat na sběrnici 1-wire
Zdroj.: [22]

Při čtení dat master nejprve nastaví sběrnici na nízkou úroveň a čeká minimálně 1 µs. Poté sběrnici uvolní a čeká 9 µs. Master přečte sběrnici, její stav udává přečtený bit, buď logická nula, nebo logická jednička.

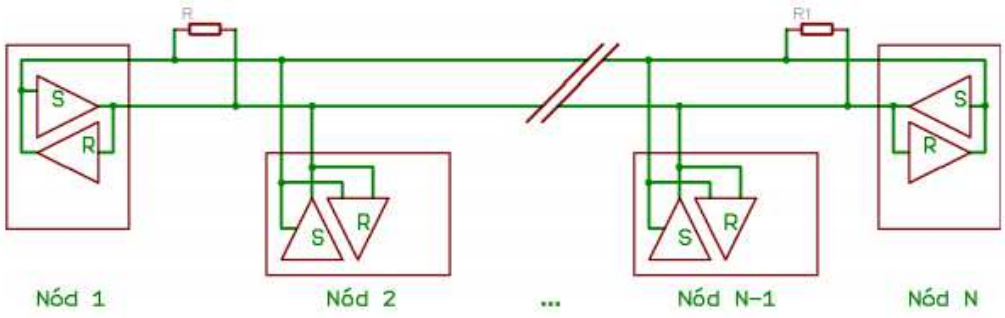
Každé zařízení na sběrnici 1-Wire má v sobě paměť ROM, která obsahuje 64bitové unikátní číslo, pomocí kterého je možné od sebe jednotlivé zařízení odlišit. Unikátní číslo se skládá z typu zařízení, sériového čísla a CRC kódu. Pokud je na jedné sběrnici připojeno více zařízení, je třeba po reset pulsu vyslat příkaz „Match ROM“, pak 64bitový kód zařízení a poté se posílají příkazy.

3.4.2 Sběrnice RS485

Specifikace RS-485 (TIA/EIA-485-A) určuje přesné elektrické vlastnosti vysílačů a přijímačů. Specifikace zmiňuje specifikace vztažené ke kabeláži a terminaci přenosových linek – nespecifikuje však zapojení konektorů nebo softwarový protokol, jako tomu je například u RS-232. Síť RS-485 muže mít až 32 připojených zařízení, kde každé zařízení reprezentuje zařízení o vstupní impedanci 12kΩ. V případě, že v síti využijeme vysoko impedanční zařízení, může mít taková síť až 256 prvků na jeden segment. Přenosová trasa ve specifikaci RS-485 muže být dlouhá až 1219 metrů anebo je schopná přenášet data rychlostí až 10Mbps (tyto podmínky nemohou být splněny současně). Při přenosové rychlosti 90kbit/z muže být přenosová trasa dlouhá až 1219 metrů; při 1Mbit/s až 121 metrů; a při 10Mbit/s se maximální délka přenosové trasy zkracuje na 15 metrů. V případě, ˇze budeme mít aplikaci, která vyžaduje více připojených zařízení nebo přenosy dat na delší vzdálenosti, lze na RS-485 přenosovou trasu nainstalovat opakovačce, které budou sloužit k regeneraci přenášených signálů.

Topologie sítě RS-485 je důvodem, proč je v dnešní době RS-485 specifikace tak oblíbeným a široce využívaným prvkem. Tato specifikace umožňuje připojit na jednu sítovou infrastrukturu hned několik vysílačů a přijímačů. [24]

Linka **RS485** je tvořena symetrickou dvojicí vodičů označovaných **a, b,** nejlépe krouceným vedením. V klidovém stavuje vodič a kladnější než vodič b. Provedení linky má být ve tvaru linie s krátkými odbočkami. Na obou koncích linky má být připojeno impedanční zakončení. Linka RS485 je poloduplexní, takže po stejném vedení se data vysílají i přijímají. Proto je nutné přepínat směr komunikace na vysílání nebo příjem. [25]



Obrázek č. 6: Topologie sítě RS485
Zdroj: [25]

Jak již bylo zmíněno, dle vstupní impedance připojených zařízení můžeme připojit maximálně 32 resp. 256 zařízení a s použitím RS-485 zesilovačů můžeme rozsah takto navrhované sítě zvýšit až na několik tisíc připojených nodů. Na obrázku č. 6 je zobrazena všeobecná topologie sítě RS-485. Kde je N zařízení připojeno do sítě typu multipoint. Pro delší vzdálenosti a při požadavku na vyšší rychlosti přenosu dat jsou ukončovací rezistory o jmenovité hodnotě 100Ω (terminátory) nezbytnou podmínkou na obou koncích přenosové soustavy tak, aby se eliminovaly všechny případné odrazy. Sít' typu RS-485 musí být navržena jako průběžná linka s odbočkami – není možné navrhovat sít' ve hvězdicové topologii z důvodu nemožnosti provedení řádné terminace, díky čemuž by došlo k razantnímu snížení kvality přenášeného signálu. [24]

3.4.3 sběrnice RS232

Sběrnice RS232 slouží k seriové komunikaci mezi dvěma zařízeními. Používá se pro připojení zařízení komunikujících maximální rychlostí 115.2 kBd na vzdálenost maximálně 15m. . Pro komunikaci se využívá dvou datových linek, RxD a TxD, přičemž každá linka je jednosměrná. Napěťové úrovně se obvykle pohybují v rozmezí pro log. 0, 3 V až 15 V a pro log. 1–3 V až – 15 V. Kromě vodičů pro přenos dat - RxD a TxD obsahuje ještě další vodiče pro řízení toku dat. [23]

Samotný komunikační standard, resp. jeho poslední varianta RS-232C, je využíván již od roku z roku 1969 [23]. Dříve měl pro komunikaci po lince RS232 téměř každý osobní počítač vyveden pomocí konektoru D-Sub typu DE-9 M (canon). RS232 je stále živý průmyslový snadnard využívaný v mnoha technických aplikacích, u osobních počítačů však ustoupil univerzálnějšímu USB rozhraní a dnes se již témeř nevyužívá.

K připojení realizovaného přístupového systému k počítači nebude na fyzické úrovni sběrnice RS232 využito. Převodníky pro komunikaci PC s mikrokontrolery jsou dnes nejčastěji vybaveny USB konektorem a přítomnost připojení k COM portu simulují.

3.5 Milníky ve vývoji přístupového systému

Vzhledem k tomu, že systém lze rozdělit na dílčí části a ty postupně uvádět do provozu, bude realizace přístupového systému probíhat v několika na sebe navazujících fázích. Nejdříve bude systém otestován na jednoduchém programu, jehož jediným cílem bude přečíst adresu iButtonu, poté budou po částech takto doplňovány a ověřovány další funkce. Pro každé rozšíření systému bude stanoven konkrétní cíl, milník projektu.

1. milník – systém umí přečíst a zobrazit adresu iButton čipu
2. milník – systém umí vyhodnotit, zda má přiložený iButton povolen přístup, disponuje 4 čtečkami a hlavním řídícím modulem, povolení či zamítnutí přístupu je signalizováno u každé čtečky zelenou / červenou LED, stejná signalizace je u hlavního modulu
3. milník – systém komunikuje s PC - po přiložení iButton čipu odešle do PC adresu iButton čipu a informaci, zda byl přístup povolen nebo zamítnut
4. milník – adresy povolený iButton čipů a přístupová práva jsou uloženy v EEPROM paměti hlavního modulu, z počítače lze měnit nastavení přístupových práv
5. milník – je realizována počítačová aplikace realizovaná v prostředí Promotic, která iformuje o aktuální dění na čtečkách, zaznamenává historii přístupů do jednotlivých místností, umožňuje měnit nastavení přístupových práv a spouští alarm při mimořádných událostech
6. milník – k realizovanému systému je vytvořen výukový materiál

Obrázek č. 7 : Znázornění postupu vývoje přístupového systému
Zdroj: vlastní

3.6 Vývojová linka

Realizace každé dílčí části do provozu bude probíhat po vývojové lince v souladu s vývojovým cyklem pro tvorbu projektu, tedy od návrhu konceptu řešení nové komponenty, přes návrh a realizaci HW a SW, oživení, testování, odstranění chyb a uvedení nové části do provozu.

4. Návrh hardware

4.1 Příprava potřebného hardware

Hardware k realizaci 1. milníku
První milník řeší systém realizující funkci přečtení ID přiloženého iButtonu a zobrazení přečtené hodnoty na displeji. Po stránce hardwaru k realizaci tohoto kroku vystačí jeden mikrokontroler, grafický displej a obvod pro realizaci 1-Wire sběrnice. Téměř vše, co budeme po stránce HW potřebovat poskytne vývojová deska DM164134.
Jako vstupní zařízení budeme k desce připojovat 1-wire čtečku, výstupem bude na desce integrovaný OLED displej.

Hardware k realizaci 2. – 5. milníku
K realizaci dalších milníků budeme muset vytvořit již téměř kompletní zapojení. K jeho realizaci budou použity dva mikrokontrolery. Jeden mikrokontrolere bude zodpovědný za přečtení iButtonu čipu a indikaci jeho přístupových práv ***/plní funkci čtečky, je v roli Slave jednotky/*** a druhý mikrokontroler bude provádět vyhodnocení přístupových práv ***/plní funkci řídícího modulu, je v roli Master jednotky/***.

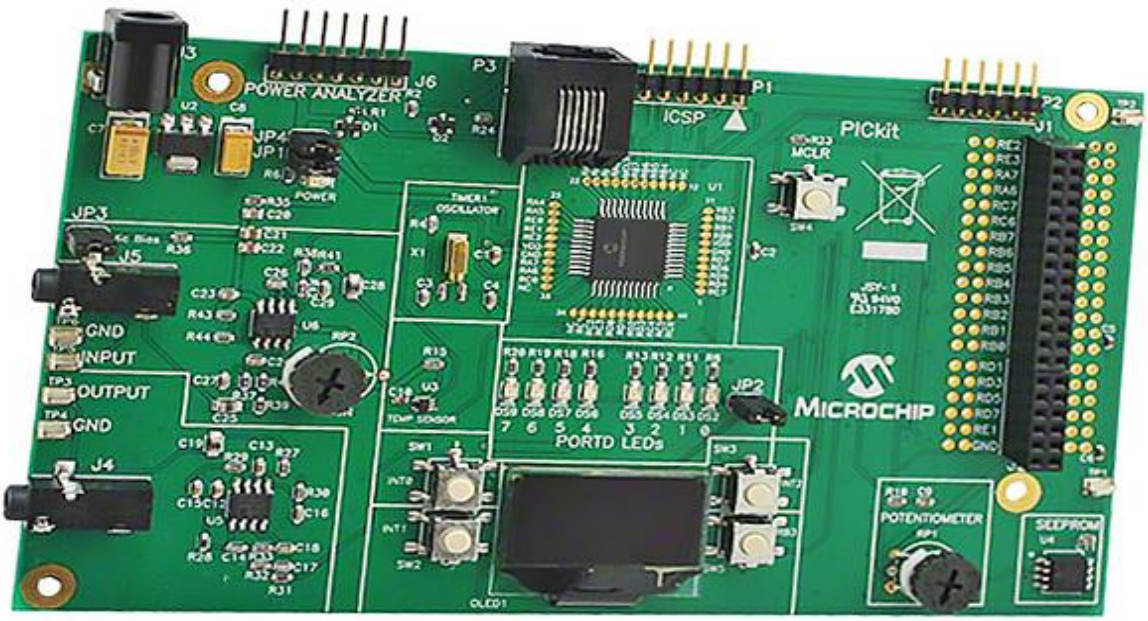
Komunikace mezi řídícím modulem a modulem čtečky bude realizována prostřednictvím diferenciální sběrnice RS485.
Pro realizaci hlavního modulu bude využita vývojová deska s procesorem PIC18LF46K22, tento mikrokontroler disponuje dvěma rozhraními UART.
Jako modul čtečky poslouží mikrokontroler PIC16F18323 disponující UART rozhraním. Pro převod dat z TTL úrovně mikrokontrolerů /UART/ na napěťové úrovně linky RS485 poslouží převodník SN75176.
Stejně tak pro komunikaci mezi hlavním modulem a počítačem bude třeba využít převodníku UART/RS232, například převodník využívající čip FTDI nebo PL-2303.

Potřebné komponenty:

- vývojová deska DM164131
- 4 x mikrokontroler PIC16F18323
- 5 x převodník SN74176
- 21 x rezistor 510 R
- 4 x rezistor 4k7
- 4 x rezistor 270 R
- 5 x zelená LED
- 5 x červená LED
- 5 x modrá LED
- 1 x žlutá LED
- 4 kontaktní čtečky čipů iButton
- vodiče
- 6 čipů Maxim iButton DS1990

4. 2Vývojová deska DM164134 – hlavní funkce a parametry

- Mikrokontroler PIC18FL46K22
- 128x64 organický LED displej (SPI)
- 32.768 kHz externí oscilátor (Timer1)
- Možnost filtrování analogového vstupu
- PWM výstupní filtrování na pinu RC2
- 4 tlačítka pro uživatelské rozhraní
- 1 přepínač MCLR
- 8 LED mapovaných na PORTD
- Potenciometr
- 1024 KB Serial EEPROM
- Teplotní čidlo MCP9700
- PICtail ™ pro rozšiřující desku
- 6-kolíkové ICSP ™ programovací rozhraní
- 6-kolíkové rozhraní pro PICkit sériový analyzátor
- RJ-11 konektor pro ICSP programování



Obr. č. 8: vývojová deska Microchip DM164131

Zdroj: [17]

4.3 MCU pro hlavní modul /Mikrokontroler PIC18LF46K22 /

Pro řízení celého systému byl vybrán mikrokontroler PIC18LF46K22, který patří mezi osvědčené osmibitové mikrokontrolery. Tento mikrokontroler disponuje vhodným hardwarovým vybavením vyhovujícím zamýšlené aplikaci a jeho architektura je optimalizována pro programování v jazyce C. PIC18LF46K22 splňuje nutná kritéria pro master jednotku, tedy podpora dvou UART rozhraní a dostatečně velká Flash / EEPROM paměť. Mikrokontroler je osazen na vývojové desce DM164131. Mikrokontroler PIC18LF46K22 s popsanými výstupy je na obrázku č. 10.
Základní vlastnosti tohoto mikrokontroleru jsou pak uvedeny v tabulce č. 2.

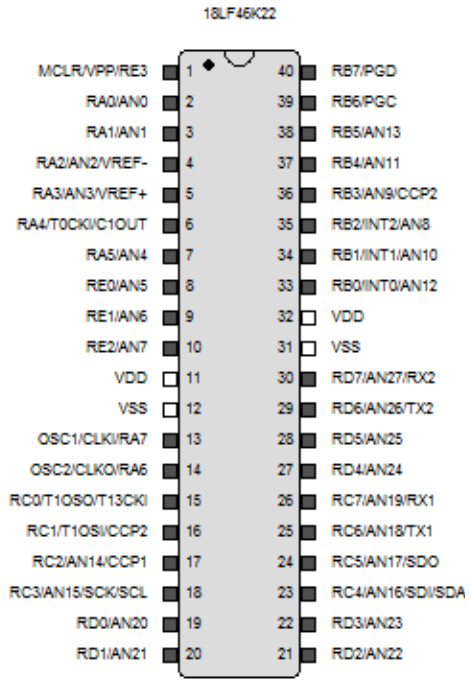
4.4 MCU pro moduly čteček / Mikrokontroler PIC16F18323 /

PIC16F18323 je pokročilý 8-bitový mikrokontroler disponující UART rozhraním (EUSART), vyznačující se nejširším množstvím funkcí z řady PIC16F183xx, zároveň je relativně levný a vyznačuje se nízkou spotřebou [18].

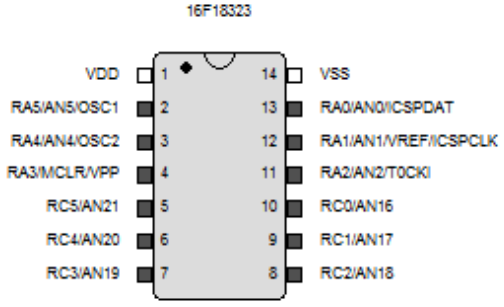
Tento mikrokontroler byl vybrán jako velmi vhodný pro realizaci modulu čtečky. Vyrábí se v pouzdrech DIP14, SOIC a TSOP. Byla vybrána dostupnější varianta v pouzdru SOIC, jejíž předností kromě nižší ceny také extrémně nízká spotřeba (XLP) [19].

Firma Microchip uvádí pro své výrobky spadající do portfolia eXtreme Low Power (XLP) PIC® spotřebu do 30 μA /MHz při běžném provozu a 9 nA ve sleep režimu.

Vybrané funkce a parametry mikrokontroleru PIC16F18323 jsou shrnuty v tabulce č. 2.



Obr. č. 10: Mikrokontroler 18LF46K22 s označenými vývody
Zdroj : vlastní, simulační prostředí Flowcode



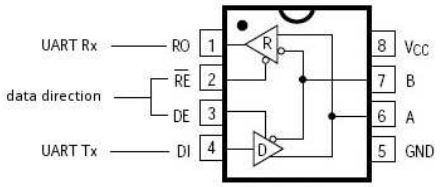
Obr. č.11: Mikrokontroler 16F18323 s označenými vývody
Zdroj : vlastní, simulační prostředí Flowcode

4.5. Převodníky sběrnice RS485 / převodník SN75176/

Jelikož mikrokontrolery budou připojeny ke sběrnici RS485 pracující s diferenciálními signály, je třeba použít signálových převodníků na úroveň kompatibilních s TTL logikou mikrokontrolerů. Pro buzení sběrnice pracující na fyzickém rozhraní linky RS 485 bude zvolen převodník SN75176 od firmy Texas Instruments v 8-pinovém pouzdře DIP.
Zapojení výstupů obvodu SN75176 je na obr. č. 9.

Základní vlastnosti obvodu SN75176

- Fyzický standart ANSI EIA/TIA-422-B
- Napájecí napětí 4,5 až 5,5V
- Odolný proti rušení
- Samostatná volba směru
- 3 stavový výstup
- Vstupní impedance 12KΩ
- Možnost připojení 32 obvodů na linku



Obr. č. 12: zapojení vývodů obvodu SN75175
Zdroj: [21]

4. 6 Kabel s převodníkem UART/USB /převodník PL2303/

Pro přenos dat z hlavního modulu do počítače bude využit 4-pinový kabel PL 2303 (produkt firmy Kuongshun Electronic) s konektorem USB. Jedná se o kabelový převodník s integrovaným obvodem PL 2303 HX uloženým přímo v plastovém pouzdře USB konektoru. Po připojení do počítače se převodník připojí k virtuálnímu COM portu.

Obr. č. 13: Převodník UART TTL na RS232 - použit převodník 75176
Zdroj: [26]

Převodník PL 2303

Základem převodníku je obvod 2303HX od Tchajwanského výrobce, firmy Prolific Technology Inc.
Jde se o miniaturní integrovaný obvod v pouzdře SSOP28 a slouží pro připojení zařízení s UART

rozhraním k hostitelskému zařízení standardu RS232. Umožní připojenému zařízení plně duplexní asynchronní sériovou komunikaci s hostitelem prostřednictvím libovolného USB.
Obsažený driver umožňuje simulovat tradiční COM port na většině operačních systémů. PL-2303HX je určen výhradně pro mobilní a vestavěná řešení. Obvod byl navržen s ohledem kompaktní rozměr a velmi malou spotřebu energie v každém provozním režimu. Byl navržen pro 3,3V logiku, ale je tolerantní i k 2,5V a 5 V logice.

Alternativně jde použít také převodník s čipem CP2102 od firmy Silicon Labs nebo FT2302 od výrobce FTDI. [26]

Parametr	**Mikrokontroler PIC16F18323**	**Mikrokontroler PIC18LF46K22**
Velikost paměti	Flash – 3,5KB EEPROM – 265 Byte	Flash – 64 KB EEPROM – 1024Byte
Rychlost procesoru	8 MIPS	16 MIPS
Rozhraní pro digitální komunikaci	1 - UART 1 - SPI 1 - I2C	2 - UART 2 - SPI 2 - I2C
Podpora programování ICSP	ANO	ANO
Rozsah provozního napětí	2,3 V – 5 V	1.8V-3.6V
Počet vývodů	14	40
Spotřeba energie v aktivním módu	30 μA / Mhz	45 μA / Mhz
Spotřeba energie v klidovém módu	9 nA	- Sleep mode: 100 nA, typical - Watchdog Timer: 500 nA, typical

Tabulka č.2 – srovnání vybraných parametrů použitých mikrokontrolerů
Zdroj: vlastní zpracování dle [27], [28]

5. Příprava realizace softwarové části

5.1 Vývojové prostředky

5.1.1 Programování mikrokontroleru, programátor PICkit 3

Vybraný mikrokontrolér lze programovat sériově pomocí SPI rozhraní. Tento způsob programování bývá také označován zkratkou ICSP (In Circuit Serial Programming). Výhodou ICSP programování je možnost nahrát program do mikrokontroleru přímo ve vývojovém prostředí MPLAB-X IDE, což velmi usnadňuje a urychluje vývoj programu. Při ISP programování se používá pětice vodičů. Jedná se o vodiče označené PGC, PGD, GND, VCC a MCLR.

Programátor PICKIT

Pro samotné provedení naprogramování byl využit programátor PICKIT3.
Programátor PICKit je vyvíjen pro obvody společnosti Microchip a nyní již existuje ve své třetí verzi. Podporuje programování pamětí EEPROM a velkého množství mikroprocesorů od řady PIC10F, po PIC32 až dsPIC. Originální programátorem lze programovat výhradně obvody společnosti Microchip.[m1]

Vzhledem k tomu, že Microchip nabízí dokumentaci s jeho schématem, přeložený firmware pro procesor a ovládací software pro PC volně ke stažení, není problém si tento programátor postavit, nebo si koupit některý z existujících klonů. U klonů originálního PICkitu za cenu zjednodušení a

Obr. č. 14 Programátor PICkit3
Zdroj: [m2]

snížení ceny produktu nemusí být vždy podporovány všechny funkce, ale může mít oproti originálu i některé funkce navíc, jako je podpora procecorů AVR .

K programování obvodů byl využit originální PICkit 3 poskytnutý vedoucím práce.
Velkou výhodou programátoru PICKIT 3 je v přímá podpora ve vývojovém prostředí MPLAB-X IDE. Uživatel po napsání kódu programu může pouze jedním stiskem přeložit svůj projekt a zároveň je ihned vytvořený program naprogramován. Programovaný obvod se může programovat již osazený za předpokladu, že jsou vyvedeny vývody pro sériové programování. Další velkou výhodou je zobrazení obsahu pamětí a registrů, a to přímo ve fyzickém mikroprocesoru. Díky tomu můžeme efektivně ladit program a hned jej sledovat v reálném čase, nebo lze krokovat například za účelem nalezení chyby. [x1]

5.1.2 Vývojové prostředí MP LAB

MPLAB je proprietární freeware integrované vývojové prostředí pro vývoj vestavěných aplikací na mikroprocesory PIC a dsPIC a je vyvinut společností Microchip Technology . MPLAB obsahuje vývojové nástroje jako je MPLAB IDE, MPLAB IPE, kompilátory, knihovny, demo soubory a další nástroje pro vývoj aplikací pro mikrokontrolery. [29]

MPLAB IDE

MPLAB IDE je integrované vývojové prostředí (Integrated Development Environment), poskytuje kompletní paletu nástrojů pro realizaci programů pro mikro kontroléry.

Obsahuje nástroje určený pro psaní, vývoj a ladění aplikací pro mikroprocesory PIC obsahující vedle textového editoru podporujícího zvýraznění syntaxe, záložky, definici barev a písem také debuger, manažer projektů, softwarový simulátor pro PIC12/16/17/18XXX a dsPIC, macro assembler MPASM a assembler MPLAB ASM30. Celý tento komplet je možné rozšířit o volitelné doplňky jako jsou kompilátory jazyka C nebo emulátory. [29]

MPLAB-X IDE

MPLAB-X je v současnosti nejnovější edicí MPLAB a přináší mnoho změn v celé paletě nástrojů pro vývoj mikrokontrolerů PIC. Na rozdíl od předchozích verzí modulu MPLAB IDE, které byly vyvinuty zcela vlastně, MPLAB-X IDE je založen na open source NetBeans IDE od společnosti Oracle. Toto nové řešení umožnilo velmi rychle a snadno přidávat mnoho často požadovaných funkcí a zároveň poskytuje mnohem rozšiřitelnější architekturu, která by v budoucnu ,měla přinést ještě více nových funkcí. [29]

MP LAB-X IPE

Integrované programovací prostředí (IPE) je samostatná softwarová aplikace, která poskytuje jednoduché rozhraní pro rychlý přístup ke klíčovým funkcím programátora. Aplikace IPE je vybavena pokročilými programovacími funkcemi a poskytuje bezpečné programovací prostředí. Nástroj IPE je multiplatformní aplikace, což znamená, že lze spustit nezávisle na IDE. Jelikož MPLAB-X IPE slouží především k programování (nahrání existujícího HEX souboru do mikročipu), je to malá a svižná aplikace. IPE pracuje s programátory / debuggery jako je Microchip PICkit ™ 3, ICD3 a REAL ICE. MPLAB IPE je obsažen v balíčku MPLAB® X IDE, takže po instalaci MPLAB-X IDE je automaticky také nainstalovaná. [29]

C – kompilátory MPLAB XC

S příchodem nové verze kompilátorů XC došlo ke sjednocení kompilátorů Microchip a Hitech, přičemž se použilo to nejlepší z každého. Kompilátor pro 8 bitové MCU vychází z Hi-tech kompilátoru pro PIC10/12/16 a PIC18. Pro 16 a 32 bitové kompilátory slouží jako základ Microchip C. [30]

XC8

Kompilátor MPLAB® XC8 C je plnohodnotný, vysoce optimalizovaný překladač ANSI C pro všechny 8bitové rodiny mikrokontrolerů PIC. Tento kompilátor je integrován do Microlab MPLAB X IDE, je kompatibilní se všemi debuggery a emulátory Microchip a běží na systémech Windows®, Linux® a MacOS®. [30]

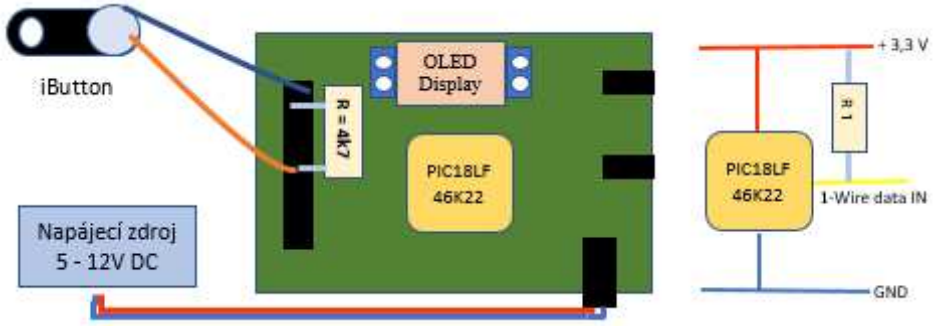
Ke tvorbě firmwaru pro mikrokontrolery bylo využito vývojového prostředí MP LAB X IDE 4.15 a programovacího prostředí MP LAB IPE 4.15.

5.2 Nástroje pro simulaci

Pro vytvoření návrhu zapojení a pro simulaci běhu programů v navrženém systém bylo využito prostředí Proteus. Jde o simulační program zaměřen na simulaci individuálních mikrokontrolerů (VSM = Virtual System Modelling). Simulační prostředí zahrnuje funkce pro kreslení schématu, modul ProSPICE pro analogové, digitální a smíšené A/D simulace, 750 modelů různých mikroprocesorů, celou řadu virtuálních přístrojů (osciloskop, logický analyzátor, funkční generátor atd.) a program pro návrh DPS. Výrobce nabízí program i v bezplatné verzi v Trial licenci.

6. Vlastní realizace přístupového systému

Šestá kapitola se věnuje vlastní realizaci přístupového systému, po jednotlivých dílčích na sebe navazujících částech, tak jak byly definovány v podkapitole 3.4 Milníky ve vývoji přístupového systému. Tato kapitola společně se zdrojovými souboru v přílohové části diplomové práce zároveň představuje úplnou technickou dokumentaci realizovaného sytému.



Obr. č. 15: Realizace jednoduché 1-wire čtečky s využitím vývojove desky DM164131
Zdroj: vlastní

6.1 Realizace milníku č.1. – přečtení a zobrazení adresy iButton čipu

Hardware

Příprava hardwaru pro realizaci 1. milníku je velmi jednoduchá, bude využita vývojová deska s procesorem PIC18LF46K22. K realizaci 1-Wire čtečky postačí k vývojové desce připojit dva vodiče a pull-up rezistor 4k7.

Kusovník použitých součástek:

- Vývojová deska DM164131
- Rezistor 4k7
- Vodiče
- Čip Maxim iButton DS1990

Zapojení

Na konektor PICtail byly přivedeny vodiče pro čtení dat z identifikačního čipu iButton, jeden na datový vstup, druhý na GND. Jako datový vstup pro 1-wire čtečku byl využit vstup RB04. Aby byla v klidovém stavu pevně nastavena úrovně log1, byl vstup RB04 připojen k napětí 3,3V přes pull-up rezistor 4k7. Připojení vodičů k PICtail konektoru na vývody RB04 a GND je na obrázku č. 7.

Software

Prvním krokem je vytvoření funkce pro přečtení adresy iButtonu a zobrazování přečtených dat na OLED displeji. Výhodou je, že na stránkách výrobce vývojové desky DM164134, společnosti Microchip, je několik programů s demo soubory včetně ovladače OLED driver, který umožnuje práci s OLED displejem. [y3] Zbývá vyřešit přečtení adresy iButtonu po 1-wire sběrnici.

Pro vytvoření programu pro iButton čtečku využijeme zdrojové soubory demo projektu oled.X. Do zdrojového souboru main.C doplníme funkce pro čtení dat po one-wire sběrnici a jejich zobrazení na displeji. Ukázka doplněných funkcí je níže na obrázku č.8, celý zdrojový kód je pak v přílohové části práce.

Detekce a přečtemí adresy iButtonu:

```
unsigned char OWReset(void)
{
unsigned char DallasPresence = 0;                // Detect Dallas fob

        TRISB=TRISB&0xEF;                                       //Setting Pin7 in PORTB to be output
        PORTBbits.RB4 = 0 ;                                     //Setting value of Pin4 in PORTB to be zero
        __delay_us(480);                                //Delay the code for 480 microseconds
        TRISB=TRISB | 0x10;                                     //Setting the direction of Pin4 in PORTB to be input
        __delay_us(70);                                         //Delay the code for 70 microseconds
        DallasPresence = PORTBbits.RB4 ;        //Reading the input value from Pin4 in PORTB to assign it as DallasPresence value
        __delay_us(410);                                //Delay the code for 410 microseconds

        return DallasPresence;                  //return the value of DallasPresence
}

unsigned char OWReadBit(void)
{
unsigned char Dbit = 0;                                 // Dallas bit result

        TRISB=TRISB&0xEF;                                       //Setting Pin7 in PORTB to be output
        PORTBbits.RB4 = 0 ;                                     //Setting value of Pin4 in PORTB to be zero
        __delay_us(6);                                          //Delay the code for 6 microseconds
        TRISB=TRISB | 0x10;                             //Setting the direction of Pin4 in PORTB to be input
        __delay_us(9);                                          //Delay the code for 9 microseconds
        Dbit = PORTBbits.RB4;                           //Reading the input value from Pin4 in PORTB to assign it as Dbit
value
        __delay_us(55);                                         //Delay the code for 55 microseconds
        __delay_us(10);                                         //Delay the code for 10 microseconds

        return Dbit;                            //return the value of Dbit
}
```

Původní demo projekt oledX i upravený kód je v přílohové části práce (příloha č. 1) a dále je uložen v adesaři *//zdrojove_soubory/precteni-zobrazeni_oled/oledX/* na přiloženém CD.

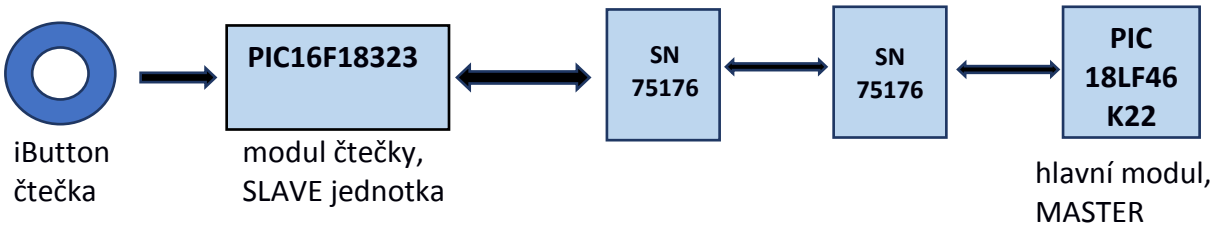
6.2 Realizace milníku č.2 – vyhodnocení povolených iButton čipů

Cílem 2. milníku je porovnávat přečtenou adresu iButton čipu se seznamem povolených adres iButton čipů a vyhodnotit výskyt shody.

6.2.1 Použitý hardware

K realizaci byly použity dva mikrokontrolery. Jeden mikrokontrolér je zodpovědný za přečtení iButtonu čipu a indikaci jeho přístupových práv /plní funkci čtečky, je v roli *Slave jednotky*/ a druhý mikrontroler bude provádět vyhodnocení přístupových práv /plní funkci řídícího modulu, je v roli *Master jednotky*/.

Komunikace mezi řídícím modulem a modulem čtečky byla realizována prostřednictvím diferenciální sběrnice RS485. Pro realizaci řídícího modulu byla využita vývojová deska s procesorem PIC18LF46K22. Jako modul čtečky posloužil mikrokontroler PIC16F18323. Pro převod dat z TTL úrovně mikrokontrolerů /UART/ na napěťové úrovně linky RS485 poslouží převodník SN75176.



Obrázek č.17 Signálové schéma pro realizaci 2. milníku přístupového systému
Zdroj: vlastní

6.2.2 Software

Pro mikrokontrolery byl vytvořen firmware. Mikrokontroler modulu čtečky je v roli Slave, čte po 1-wire sběrnici adresu iButtonu a odesílá ji do mikrokontroleru řídící jednotky, který je v roli Masteru. Na základě povelů z Masteru provede indikaci povolení vstupu nebo odepření vstupu. /**program iButton_slave.c**/

Mikrokontroler hlavní jednotky je naprogramován tak, aby vyhodnocoval požadavky vstup přicházející od jednotlivých čteček. /**program iButton_master.c**/

Činnost programů je popsána vývojovými diagramy uvedenými v příloze č. II. , finální verze programů pro hlavní mikrokontroler jsou pak v příloze III a zdrojový kód programů pro mikrokontroler čtečky je v příloze IV.

Programy pro hlavní mikrokontroler (PIC18LF46K22)

iButton_master.h

Hlavičkový soubor obsahuje direktivy s přiřazením portů a dalších nastavení definovaným makro výrazům. Z obrázku č. 18 je patrné nastavení UART1 portovaného na RC6/RC7 a nastavení

```
// UART1 - PC:
#define UART1_TX        PORTCbits.RC6
#define UART1_RX        PORTCbits.RC7

// UART2 - RS485:
#define UART2_TX        PORTBbits.RB6
#define UART2_RX        PORTBbits.RB7
#define Rizeni_toku     LATB5           //H=Tx, L=Rx
```

Obrázek č.18: iButton_master.h, direktivy s nastavením UART1 a UART2
Zdroj: vlastní

UART2 na porty RB6 a RB7.

iButtom_master_iButtons.h

```
//############################## Kody iButton s opravnenim - zadano v HEX

const char iButtons_code_array[] =
{
    0x01,0xA1,0x44,0xCF,0x01,0x00,0x00,0xA8,    //ID iButton 1 (zleva F
```

Obrázek č.19: iButton_master_iButtons.h, pole s iButton čipy s oprávněním k přístupu
Zdroj: vlastní

iButton_master.c

Soubor iButton_master.c je hlavním programem ve kterém jsou definovány všechny funkce a obsahuje hlavní proceduru. Hlavní provozní smyčka je na obrázku č.22. Běh programu se signalizován blikající stavouvou LED.

```
//#########################################################################################
void Vyhodnot_data_od_UART2()
{
    Flag_shoda=0;

    if(Buffer_Rx_UART2[0]=='S' && Buffer_Rx_UART2[10]=='E'  )          //Pokud vysila Slave a je v poradku startovaci a
    {
        ID_Slave=Buffer_Rx_UART2[1];                                   //Tady je ulozeno cislo ctecky
        if(Porovnej_kody_iButton()==nalezena_shoda) Flag_shoda=1;      //Porovna se prijate ID iButton s temi, ulozenym
        Vyhodnot_vystup(Flag_shoda);
        Flag_odeslat_odpoved=1;
    }
}
//#########################################################################################
```

Obrázek č.20: Definice funkce Vyhodnot_data_od_UART().
Zdroj: vlastní

Pokud dojde k vyvolání přerušení od UARTU2 (RS485), stavová LED nebliká, ale svítí a je volána funkce ***Vyhodnot_data_od_UART2()***. Definice funkce ***Vyhodnot_data_od_UART2()*** je na obrázku č. 20.

Podívejme se nyní podrobněji na funkci ***Vyhodnot_data_od_UART2()***
Tato funkce nejdříve rozdělí příchozí řetězec na podřetězce (UART[0] - UART [10]). V případě, že řetězec odpovídá očekávanému formátu, je podřetězec obsahující adresu iButtou za pomoci funkce **Porovnej_kody_iButton()** /viz obrázek č.21/ porovnán s obsahem pole s uloženými adresami povolených iButton čipů (uloženy v iButton_master_iButtons.h).

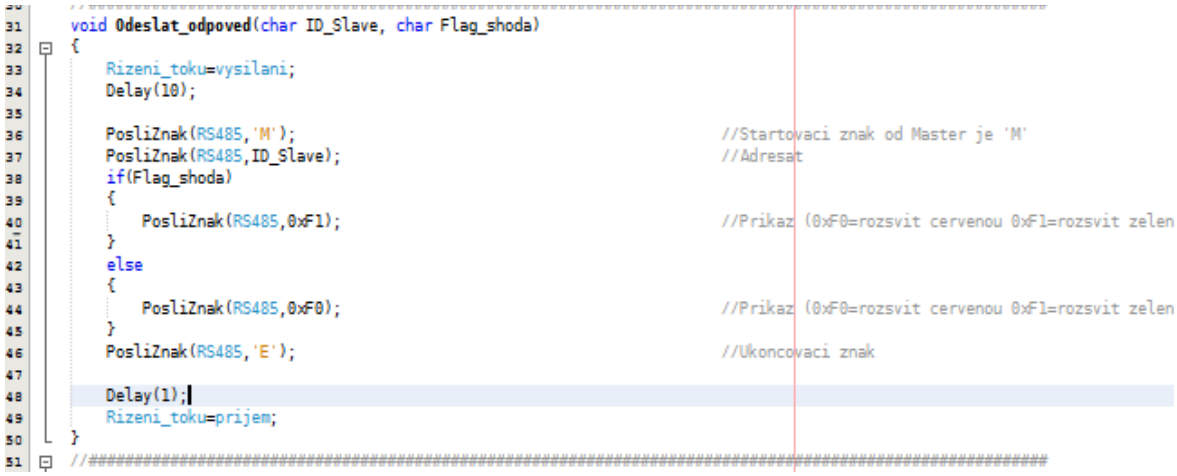
```
bit Porovnej_kody_iButton()
{
   char Buff_pointer=2;
   char Page_pointer=0;

        do{
            if(Buffer_Rx_UART2[Buff_pointer]==iButtons_code_array[(Buff_pointer-2)+Page_pointer])  //Prohled
                {
                    if(Buff_pointer==9) { return 1; } else { ++Buff_pointer; }
                }
                else
                {
                    Page_pointer=Page_pointer+8; Buff_pointer=2;
                }

           }while(Page_pointer<=72);

    return 0;
}
```

Obrázek č.21: funkce Porovnej_kody_iButton ().
Zdroj: vlastní

V případě nalezení shody je nastaven Flag_shoda =1.
Následně je v volána funkce **Vyhodnot_vystup(**Flag_shoda**).** Po dokončení porovnání je nastave Flag_odeslat_vystup =1.

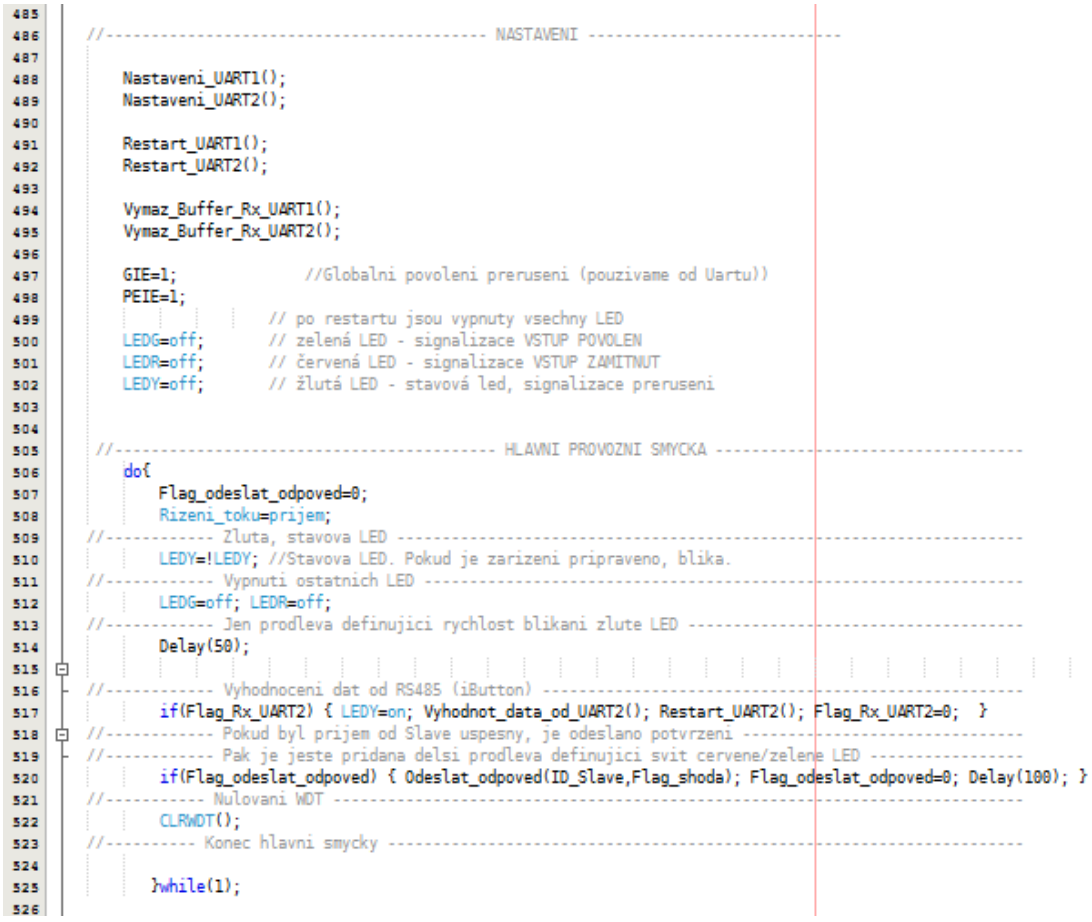
Funkce **Vyhodnot_vystup** v případě, že Flag_shoda =1 zapne zelenou LED, v opačném případě zapne červenou LED.

Vraťme se nyní do hlavní provozní smyčky. Pokud se data podařilo vyhodnotit, je volána funkce **Odeslat_odpoved (ID_Slave, Flag_shoda) .** Kde parametr ID_Slave (0x01 až 0x20) představuje číslo čtečky, ze které požadavek přišel, a které má být tedy odpověď odeslána. Funkce odeslat odpověď je na obrázku č.20. Hlavní provozní smyčka pak na obrázku č. 22.



```
31  void Odeslat_odpoved(char ID_Slave, char Flag_shoda)
32  {
33      Rizeni_toku=vysilani;
34      Delay(10);
35
36      PosliZnak(RS485,'M');                                   //Startovaci znak od Master je 'M'
37      PosliZnak(RS485,ID_Slave);                              //Adresat
38      if(Flag_shoda)
39      {
40          PosliZnak(RS485,0xF1);                              //Prikaz (0xF0=rozsvit cervenou 0xF1=rozsvit zelen
41      }
42      else
43      {
44          PosliZnak(RS485,0xF0);                              //Prikaz (0xF0=rozsvit cervenou 0xF1=rozsvit zelen
45      }
46      PosliZnak(RS485,'E');                                   //Ukoncovaci znak
47
48      Delay(1);
49      Rizeni_toku=prijem;
50  }
51  //#############################################################################################
```

Obrázek č.22: funkce Odeslat_odpoved ().
Zdroj: vlastní



```
485
486  //---------------------------------------- NASTAVENI ---------------------------
487
488      Nastaveni_UART1();
489      Nastaveni_UART2();
490
491      Restart_UART1();
492      Restart_UART2();
493
494      Vymaz_Buffer_Rx_UART1();
495      Vymaz_Buffer_Rx_UART2();
496
497      GIE=1;              //Globalni povoleni preruseni (pouzivame od Uartu))
498      PEIE=1;
499                      // po restartu jsou vypnuty vsechny LED
500      LEDG=off;       // zelená LED - signalizace VSTUP POVOLEN
501      LEDR=off;       // červená LED - signalizace VSTUP ZAMITNUT
502      LEDY=off;       // žlutá LED - stavová led, signalizace preruseni
503
504
505   //---------------------------------------- HLAVNI PROVOZNI SMYCKA ---------------------------
506      do{
507          Flag_odeslat_odpoved=0;
508          Rizeni_toku=prijem;
509  //------------ Zluta, stavova LED ------------------------------------------------------------
510          LEDY=!LEDY; //Stavova LED. Pokud je zarizeni pripraveno, blika.
511  //------------ Vypnuti ostatnich LED ---------------------------------------------------------
512          LEDG=off; LEDR=off;
513  //------------ Jen prodleva definujici rychlost blikani zlute LED ----------------------------
514          Delay(50);
515
516  //------------ Vyhodnoceni dat od RS485 (iButton) --------------------------------------------
517          if(Flag_Rx_UART2) { LEDY=on; Vyhodnot_data_od_UART2(); Restart_UART2(); Flag_Rx_UART2=0;  }
518  //------------ Pokud byl prijem od Slave uspesny, je odeslano potvrzeni ----------------------
519  //------------ Pak je jeste pridana delsi prodleva definujici svit cervene/zelene LED --------
520          if(Flag_odeslat_odpoved) { Odeslat_odpoved(ID_Slave,Flag_shoda); Flag_odeslat_odpoved=0; Delay(100); }
521  //------------ Nulovani WDT ------------------------------------------------------------------
522          CLRWDT();
523  //---------- Konec hlavni smycky -----------------------------------------------------------
524
525        }while(1);
526
```

Obrázek č. 23: hlavní provozní smyčka programu iButton_master.c
Zdroj: vlastní, snímek obrazovky v prostředí MPLAB-X IDE.

Programy pro mikrokontroler čteček (PIC16f18323)

Pro program pro mikrokontrolery čteček byly vytvořeny zdrojové soubory **iButton_slave.c, iButton_slave.h a iButton_slave_retezce.h**

Tyto programy zůstaly na dále beze změn, při realizaci dalších milníků byl upravovány pouze programy pro hlavní mikrokontroler.

iButton_slave.h

```
//1-Wire
#define Port_1Wire  RC0

// UART + rizeni RS485:
#define UART_TX     RC4
#define UART_RX     RC5
#define Rizeni_toku   LATC3       //H=Tx, L=R
```

Obrázek č.24: iButton_master.h, direktivy s nastavením UART1 a UART2
Zdroj: vlastní

iButton_slave_string.h

V tomto souboru je direktiva ID_Slave pro nastavení čísla čtečky

```
//############################ Cislo ctecku iButton, ktere se vysila na Master jako ID ctecky:

#define     ID_Slave                0x03
```

Obrázek č.25: iButton_master_string.h , nastavení čísla čtečky
Zdroj: vlastní

iButton_slave.c

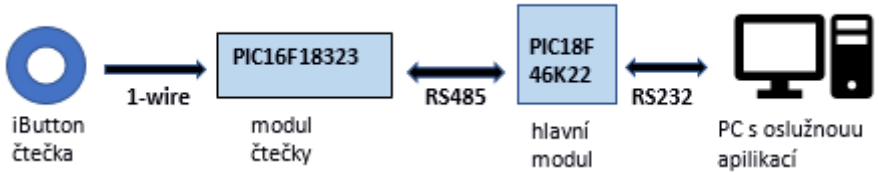
Samotné přečtení čipu po 1-wire sběrnici již bylo vyřešeno v předchozím milníku. Realizace programu pro mikrokontrolery čteček obnášela tedy především řešení komunikace mezi Master-Slave mezi čipy. Na obrázku č. 26 je funkce která řeší přijetí dat o Masteru a rozsvícení zelené LED v případě povolení vstupu nebo červené LED, pokud byl vstup zamítnut.

```
//#########################################################################################
void Vyhodnot_data_od_UART()
{
    if(Buffer_Rx_UART[0]=='M' && Buffer_Rx_UART[3]=='E'  )          //Pokud vysila Master a je v poradku startovaci
    {
        if(Buffer_Rx_UART[1]==ID_Slave)                             //Pokud souhlasi adresa Slave, provede prikaz
        {
            if(Buffer_Rx_UART[2]==0xF0) { LEDR=on; }                //Prikaz od Master - Ibutton OK - rozsvit
            if(Buffer_Rx_UART[2]==0xF1) { LEDG=on; }                //Prikaz od Master
            Delay(100);
        }
    }

}
//#########################################################################################
```

Obrázek č.26: iButton_slave.c , přijatý řetězec od Masteru a zapnutí patřičné LED
Zdroj: vlastní

6.3 Realizace milníku č.3 – připojení 4 čteček, odesílání dat do PC

Dalším cílem je realizovat odesílání dat z hlavního modulu do osobního počítače k následnému zpracování v Promotic aplikaci. A to tak, že data jsou vysílána přes UART rozhraní mikrokontroleru po sériové lince (RS232) na USB port počítače. Zvolený mikrokontroler disponuje dvěma UART rozhraními, jeden UART je využit na komunikaci s moduly čteček, druhý UART využijeme na komunikaci s počítačem. Přenos dat mezi řídícím modulem a počítačem zajistí převodník rozhraní UART (TTL , RS23) na sběrnici USB (kabel s integrovaným převodníkem).



Obrázek č. 27 Blokovové schéma HW pro realizaci 3. milníku přístupového systému
Zdroj: vlastní

6.3.1 Hardware

V nepájivém poli byl kompletně sestaven systém skládající se z hlavního modulu, čtyř kontaktních čteček a čtyř modulů pro obsluhu čteček . Dále bylo realizováno propojení všech modulů prostřednictvím sběrnice RS485. Vytvoření sběrnice RS485 si vyžádalo využití pěti převodníků SN75176 a související propojovací vodiče a rezistory. Pro odesílání dat do PC byl použit UART/USB převodník PL 2303.

6.3.2 Software

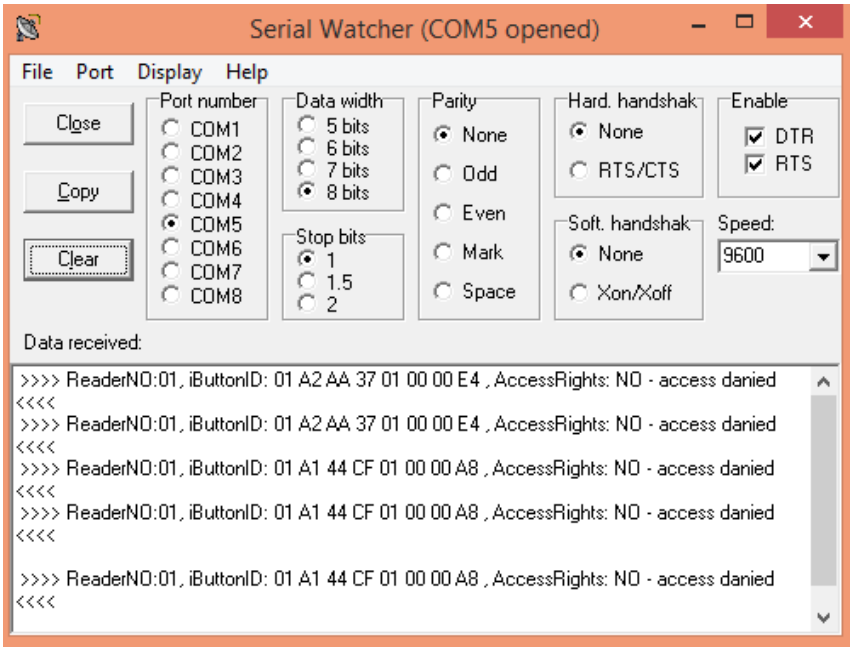
V předchozím milníku již byla vyřešena komunikace prostřednictvím UART rozhraní mezi mikrokontrolery, nebyl tedy větší problém podobným způsobem využít druhý UART pro komunikaci hlavního mikrokontroleru s počítačem.
Byl vytvořen soubor **iButton_master_retezce.h** a v souboru **iButton_master.c** byl do funkce ***Vyhodnot_vystup(char Flag_opravneni)*** doplněn skript pro odeslání dat do PC. Soubor iButton_master_retezce.h je v příloze IV, část upravené funkce ***Vyhodnot_vystup()*** je pak na obrázku č. 29.

```
//-------- Nasleduje odeslani informace do PC: --------------------
PosliText(PC,(char*)" >>>> ");      //uvodni znaky

PosliText(PC,(char*)S_Kod_Slave);     //kod ctecky ...
prevod_HEX_Ascii(Buffer_Rx_UART2[1]);
PosliZnak(PC,MSB); PosliZnak(PC,LSB); //... odeslan na PC v ASCII


PosliText(PC,(char*)S_Kod_iButton);   //kod iButton

char pozice=2;
do{                                   //odesilani kodu iButton v ASCII formatu
   prevod_HEX_Ascii(Buffer_Rx_UART2[pozice]);
   PosliZnak(PC,MSB); PosliZnak(PC,LSB); PosliZnak(PC,' ');
}while(++pozice<10);

PosliText(PC,(char*)S_Opravneni);     //Opraveni NE
PosliText(PC,(char*)S_Ne);

PosliText(PC,(char*)" <<<< ");      //ukoncovaci znaky
```

Obrázek č. 28 Blokovové schéma HW pro realizaci 3. milníku přístupového systému
Zdroj: vlastní

6.3.3 Testování

Pro testování odesílání dat z mikrokontroleru do počítače byla využita aplikace Serial Watcher.



Obr. č. 29 : Terminálové okno aplikace SeriálWatcher s přijatými informacemi o přiložených iButton čipech na čtečkách

Zdroj.: Vlastní, aplikace

6.4 Realizace milníku č. 4. – nastavení a editace přístupových práv z počítače

Realizací předchozích milníků byl vytvořen po HW stránce kompletní kontaktní přístupový systém. Systém je však schopen pouze demonstračního provozu, neboť všechny adresy iButton čipů jsou uloženy „napevno“ v programu. Následující část se zabývá úpravou programu hlavního modulu, aby bylo možno přidávat do systému další iButton čipy a měnit nastavení přístupových práv.

6.4.1 zápis iButton klíčů a přístupových práv do EEPROM paměti

Mikrokontrolér PIC18LF46K22 využitý na realizaci hlavního modulu disponuje 1024 Byte EEPROM pamětí a nabízí se tedy možnost ukládat do datové paměti jednotlivé adresy iButton klíčů. Jelikož na každou adresu potřebujeme 8 Byte, dále v 1 Byte budou uložena nastavení přístupových práv. Pro každý iButton klíč tedy potřebujeme 9 Byte. Velikost EEPROM paměti tedy omezuje maximální počet evidovaných klíčů na 113. Pokud by nastal požadavek na ukládání většího počtu uživatelů, lze k tomuto účelu využít Flashovou paměť. Flash paměť je sice programovou pamětí, vzhledem k tomu že daný mikrokontroler disponuje 64 kB flash pamětí, je z velké části nevyužita. Pro zamýšlený počet evidovaných adres iButton klíčů je však EEPROM paměť dostačující.

6.4.2 Úprava software

Pro zápis adres iButton klíčů do EEPROM paměti byly vytvořeny moduly **EEPROM_module.c** /pro zápis a čtení z EEPROM paměti/ a **iButton_operations.c** / pro obsluhu operací s iButton klíči, jako je zápis nového iButton klíče, editace přístupových práv a smazání iButton klíče ze systému. Dále byla upravena funkce ***najdi_iButton*** ve stávajícím programu iButton_master.

```
53 31 32 33 34 35 36 37 38 39 45        S123456789E
53 54 4f 52 45 20 4f 4b              STORE OK
```

Obr. č. 30: Test přidání nového iButton čipu s přijatým potvrzením
Zdroj: Vlastní, terminálové okno aplikace Termite

6.5 Finální úpravy a odladění

Poslední milník byl věnován závěrečným úpravám, odladění systému a vylepšením s ohledem na obecné nároky kladené na přístupové systémy po stránce bezpečnosti provozu a provozní spolehlivosti. Pro provoz přístupového systému v reálných podmínkách bude třeba vzít v úvahu i aspekty jako je zabezpečení chodu systému ze záložního zdroje při výpadku dodávek elektrické energie z rozvodné sítě a návrh zvýšení odolnosti systému vůči pokusům o úmyslné poškození systému zkratováním kontaktů snímacího pole čtečky nebo snahám o zničení čteček připojením zdrojů elektrické energie. Zkušenosti vzešlé z testovacího provozu přístupového systému budou zahrnuty do provodného výukového materiálu

7. Realizace dohledové aplikace v prostředí Promotic

Pro uživatelsky přívětivé nastavování povolených přístupů na jednotlivých čtečkách a vzdálený dohled nad fungováním systému byla vytvořena počítačová aplikace. Realizovaná počítačová aplikace představuje uživatelské rozhraní pro komunikaci s přístupovým systémem, slouží k příjmu informací o povolených a zamítnutých žádostech o vstup do jednotlivých místností a umožňuje nastavovat uživatelům, resp. iButton čipům přístupová práva.

Aplikace byla realizované pomocí nástroje Promotic od firmy Microsys.

7.1 Seznámení s prostředím Promotic

7.1.1 SCADA nástroj PROMOTIC – nejdůležitější parametry a komponenty

„Promotic je komplexní SCADA objektový softwarový nástroj pro tvorbu aplikací, které monitorují, řídí a zobrazují technologické procesy v nejrůznějších oblastech průmyslu. Je určen pro OS Windows 10/8/7/Vista/XP/XPe/2003-12Server a novější. Umožňuje efektivně vytvářet distribuované a otevřené aplikace v nejrůznějších odvětvích průmyslu a od verze 8 je možno provozovat systém Promotic také zdarma, ve freeware režimu.“ (Převzato z [3]).

V systému Promotic jsou zabudovány všechny nezbytné komponenty pro tvorbu jednoduchých i rozsáhlých vizualizačních a řídicích systémů:

- Editor aplikace s hierarchickým stromem objektů.
- Široká nabídka objektů PROMOTIC.
- Jazyk Microsoft Basic (VBScript) pro zápis algoritmů.
- Editor obrazů.
- Bohatá paleta technologických obrázků vytvořených ve vektorové SVG grafice.
- Grafické objekty – elementární a komplexní velmi obecně konfigurovatelné prvky.
- Automatická konverze obrazů do HTML a XML1 formátu.
- Systém trendů (tj. uchovávání hodnot s časovou známkou).
- Systém alarmů a operátorských událostí („eventů“).
- Podpora web technologií Internet/Intranet.
- SQL a ODBC rozhraní pro databáze.
- Zabudovaná rozhraní: XML, OPC2 , ActiveX, DDE3 .
- Komunikační ovladače pro přístup k PLC4 .
- Správa uživatelů, oprávnění a přihlašovací systém.
- INFO – informační a diagnostický systém.
- Elektronická i tištěná dokumentace.“ (Převzato z [P3])

7.1.2 Instalace a bezplatné použití systému Promotic

Užívání SCADA systému PROMOTIC je možné na základě zakoupení komerční licence, která umožní vývoj aplikací s použitím všech komponent bez jakéhokoli omezení. PROMOTIC je však také k dispozici jako freeware: PmFree - Bezplatné vývojové prostředí a runtime licence systému PROMOTIC. V tomto případě je jediným omezením velikost aplikace (počet proměnných), ale všechny standardní komponenty a technologie (včetně komunikačních ovladačů) jsou i v tomto případě plně funkční a projektant si tak může ověřit vše potřebné. [p2]

Instalační balíček s nejnovější verzi Promoticu lze stáhnout z webových stránek společnosti Microsys: https://www.promotic.eu/cz/promotic/download/download.htm
Instalační balíček obsahuje jak vývojové, tak runtime prostředí, a to včetně všech ovladačů, rozhraní, Web serveru, grafické knihovny a kompletní dokumentace. [P3]
Kromě samotné instalace je pro bezpečný provoz vytvářených aplikací vhodné provést změny v konfiguraci OS Windows dle doporučení uvedených v dokumentaci systému Promotic [p4].
https://www.promotic.eu/cz/pmdoc/Appendix/CfgOS.htm

7.1.3 Založení nové aplikace

Součástí dokumentace od firmy Microsys je i Učebnice Promotic. V jednotlivých kapitolách této učebnice je popsáno, jak tvořit jednoduchou aplikaci, která bude zaměřena na řízení kotelny. Začínající uživatelé si dle postupů popsaných v této učebnici mohou vytvořit aplikaci Kotelna a prakticky se tak seznámit se základními principů fungování SCADA systému PROMOTIC. Učebnice vychází z obsahu přednášek, které v rámci školení pořádá firma MICROSYS, spol. s r.o. a je dostupná na webových stránkách www.promotic.eu [p1]

7.1.4 Editor aplikace

Editor aplikace je základním nástrojem tvorby aplikací systému PROMOTIC, slouží k definování stromové struktury PROMOTIC objektů, jejich nastavení, definování algoritmů atd.

Zabudovaný jazyk VBScript se syntaxí Visual Basic slouží pro zápis uživatelských algoritmů v **událostním programování**, pro přístup k metodám a vlastnostem objektů systému PROMOTIC nebo jiných softwarových aplikací. Pro projektanta to představuje neomezené možnosti vývoje aplikace.

K odladění aplikace je k dispozici informační a diagnostický INFO systém. INFO systém umožňuje prohlížení všech důležitých informací za běhu aplikace. Nabízí se možnost vzdáleného ladění běžících aplikací
- v sítích Internet a Intranet přes PROMOTIC Web
- nebo programy TeamViewer, PCAnyWhere, LapLink, CarbonCopy, atd. [zdroj]

7.1.5 Objekty Promotic

Výhodou prostředí Promotik je, že při vytváření nové aplikace do projektu zahrneme pouze ty z přednastavených funkcí, které bude aplikace skutečně potřebovat. Spolu související funkce jsou pak

prezentovány v podobě Promotic objektů. Existují tak objekty, které řeší komunikaci (PmComm, PmCommMsg,…), ukládání dat (PmData, PmDatabaze,..) a grafickou podobu prezentace dat (PmPanel, PmTrend, PmAlarmEvent, …). Tvorba grafického prostředí je pak plně v rukou vývojáře. Tvůrci Promotiku připravili rozsáhlou knihovnu před konfigurovaných grafických prvků. Na tyto grafické prvky lze vytvářet datové vazby a vizualizovat data ze vstupních zařízení či z grafického prostředí odesílat data na výstupní zařízení. Ke zpracování dat lze využít vlastních skriptů napsaných v jazyce Javascript nebo VB skript.
Níže jsou stručně představeny objekty použité v realizované aplikaci.

PmPanel

Objekt reprezentuje okno aplikace s obrazem, ve kterém lze vizualizovat data systému PROMOTIC. Vlastní obsah grafického obrazu se vytváří v editoru obrazů pomocí grafických prvků, které mohou být datových vazeb napojené na data v aplikaci.

PmComm

Objekt zajišťuje komunikaci s jinými počítači (většinou s technologickými počítači typu PLC) **přes Ethernet** nebo **přes sériový port počítače** (COM1, COM2,...).

V objektu PmComm lze založit jeden nebo více objektů PmCommMsg nebo PmCommData.

PmCommMsg

Objekt slouží k definici formátu a dat jedné komunikační zprávy. Objekt je generován podle typu komunikačního protokolu nastaveného objektem PmComm. Z tohoto důvodu je vhodné objekt PmCommMsg založit až po konfiguraci objektu PmComm.

Objekt obsahuje dvě skupiny proměnných. Jedna slouží k parametrizování dat pro zaslání a druhá skupina slouží pro přijatá data z komunikace. K těmto proměnným lze přistupovat i pomocí skriptu.

PmData

Objekt PmData slouží k definici libovolného počtu proměnných různých datových typů v jednom objektu. K proměnným OBJEKTU PmData definovaným v záložce Data lze přistupovat i pomocí skriptu.

7.2 Vytváření vlastní aplikace AccessControl v prostředí Promotic

I když prostředí PROMOTIC je poměrně robustním nástrojem oplývajícím značným množstvím funkcí a je vhodné zejména pro rozsáhlé a náročné projekty v oblasti průmyslové automatizace, lze jeho předností využít i pro z hlediska počtu vstupů a zpracovávaných dat malý projekt, jako je obslužná aplikace přístupového systému.

Za účelem vizualizace aktuálního dění na čtečkách, k nastavování přístupových práv, ke sledování událostí při provozu přístupového systému a k ukládání historie přístupů do jednotlivých místností byla vytvořena obslužná aplikace AccessControl realizovaná v prostředí Promotic. Tato aplikace obsahuje objekty typů PmFolder, PmWorkspace, PmComm, PmData a PmPanel, PmAlarmEvent.

7.2.1 Zobrazení aktuálních přístupů na čtečkách

Po krátkém seznámení s nástrojem Promotik byl započat vývoj obslužné aplikace. Napřed bylo řešeno zobrazení dat, které jsou odesílány z hlavního modulu přes virtuální COM port do počítače. Po založení nového projektu byl započat vývoj statické části aplikace. Postup založení nového Promotic projektu, vytváření jednotlivých instancí objektů a jejich konfigurace není v této práci detailně popsán, neboť je již velmi srozumitelně popsán ve výukovém tutoriálu, který je součástí oficiální dokumentace, jenž je volně dostupná na webových stránkách firmy Microsys: https://www.promotic.eu/cz/pmdoc/Tutorial/TutorStart.htm

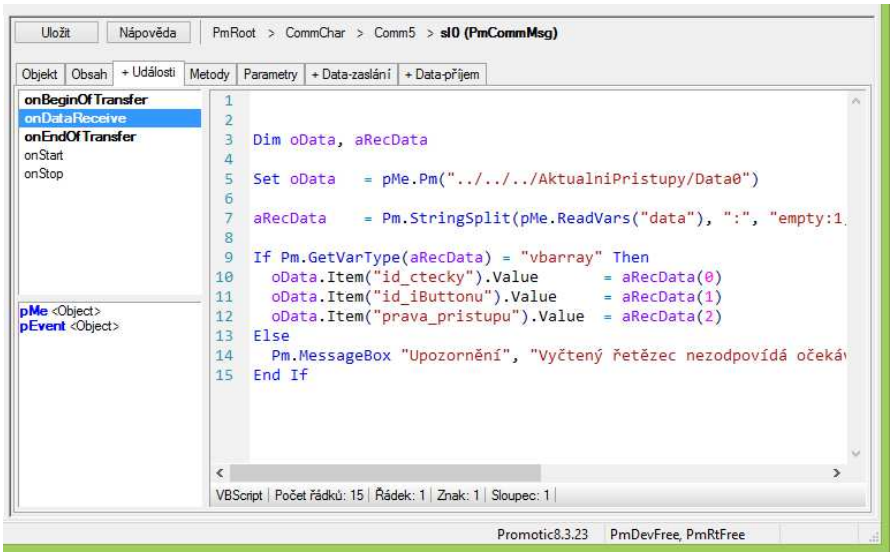
Ukládání přijatých dat

Byl vytvořen objekt iButtonData typu PmData a v něm 3 proměnné - id_ctecky, id_iButtonu, prava_pristupu. Do takto připravených proměnných budeme ukládat přijatá data z komunikace. Data přicházející v podobě jednoho řetězce budou rozdělena do jednotlivých proměnných pomocí skriptu definovaného v objektu **iButtonMsg** v události **OnDataReceive**.

Nastavení komunikace

K nastavení parametrů komunikace slouží objekt PmComm. Ve stromové struktuře Promoticu byl založen nový objekt **iButtonComm (**objekt typu PmComm**)** a u tohoto objektu v záložce Parametry zvoleny parametry komunikace. Jako komunikační ovladač byl vybrán **PmChar (serial)** a vybrán sériový port, ke kterému se bude zařízení připojovat (v tomto případě COM3).

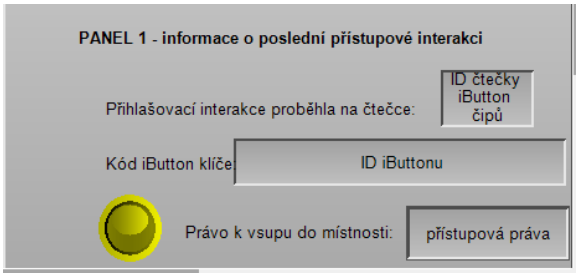
Dále pro nastavení parametrů přijímaných a odesílaných dat a jejich zpracování skriptem byl vytvořen objekt **iButtonMsg** (objekt typu PmCommMsg**).** Po příjmu dat z komunikace je automaticky vyvolán skript definovaný v události OnDataReceive. Tento skript provede rozdělení vyčteného řetězce do tří proměnných a uložení hodnot těchto proměnných do výše uvedeného objektu iButtonData. Dialogové okno objektu PmCommMsg s uvedeným VBSkriptem je na obrázku č.32.



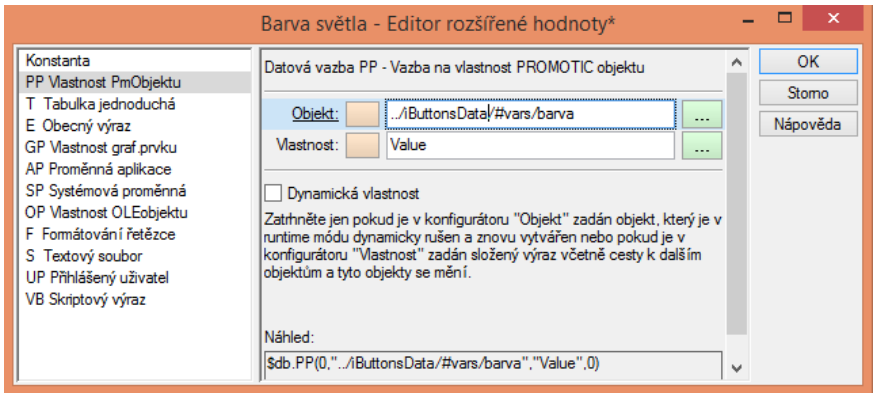
Obr. č. 32, Nastavení základních parametrů komunikace
Zdroj.: vlastní

Zobrazení dat

K zobrazení přijatých dat byl vytvořen objekt **Panel_pristupy** (objekt typu PmPanel). Tento panel zobrazuje v instanci **PANEL 1** informace aktuálně přicházející ze čteček. PANEL 1 obsahuje 3 zobrazovací pole (prvky typu PmiText).
Pro každé ze zobrazovacích polí byla v editačním okně nastavena datová vazba na příslušnou proměnou definovanou v objektu iButtonData (objekt typu PmData).



Obr. č. 33: Panel zobrazující adresu data z poslední přístupové interakce na čtečce
Zdroj.: vlastní



Obr. č. 34, Editace rozšířené hodnoty prvku Led1 – nastavení vazby na proměnnou barva
Zdroj.: vlastní

Jak je vidět na obrázku č.33., PANEL 1 obsahuje ještě grafický prvek Led1 představující dvoubarevnou LED kontrolku. Jde o grafický prvek typu PmiRasterImage, který má parametr vlastnosti barva světla nastaven na žlutou barvu (#ffff00). Vlastnost barva světla je však dynamickou vlastností. V editoru rozšířené hodnoty byla prvku Led1 nastavena datová vazba na vlastnost Promotic objektu, konkrétně na hodnotu proměnné #vars/barva.

Hodnota proměnné #vars/barva se mění v závislosti na příchozích datech. Pokud byl iButton čipu vstup do místnosti povolen, je do proměnné #vars/barva uložena hodnota #2ECC71 (tedy hex název zelenou barvu – odstín Medium Sea Green), v případě zamítnutého vstupu je nastavena hodnota #CC2E3A (červená – odstín Mahagony). Změnu hodnot provádí VBSkript spouštěný událostí onDataReceive v objektu iButtonMsg.

7. 2.2 Záznam přístupů do jednotlivých místností

Aplikaci AccessControl také zaznamená historii přístupových interakcí. Záznamy o všech úspěšných i zamítnutých přístupových interakcích jsou vedeny zvlášť pro každou místnost i pro každý iButton čip.

Každý příchozí řetězec je v okamžiku přijetí (vyvolaní události ***onDataReceive*** v objektu ***iButtonMsg***) vyhodnocen a uložen do příslušného txt souboru do složky ***Data*** v pracovním adresáři aplikace.

AccessControl.pra. Na obrázku č. 35 je VBSkript provádějící vyhodnocení.
V závislosti na čísle čtečky na niž proběhla přístupová interakce a přístupových právech je nastavena

hodnota proměnné ***cesta*** a následně sestavena úplná cesta pro uložení do txt souboru.

```
'Ukladani vsech pristupu - ve tvaru AKTUALNI DATUM A CAS, ADRESA iBUTTONU, ID CTECKY
DataVse = Array(Now, oAdresaiButton, IDctecky, PristupovaPrava)
sRow = Pm.StringFormat("%s;%s;%s;%s;", DataVse)
Pm.FileTextWrite "#app:Data/pristupy_vsechny.txt", sRow, "mode:add;"

'Nastaveni cesty pro iButtony, ktere vstupujici na ctecce 01 a pristup maji povolen.
If oData.Item("id_ctecky").Value = " >>>> R:01" AND prava  = "Ano" Then
cesta = "#app:Data/ct1_povolene.txt"

Else
.
.
.

'Ukladani pristupu rozdelene podle jednotlivych mistnosti a pristupovych prav.
DataSortovane = Array(Now, oAdresaiButton)
sRow = Pm.StringFormat("%s;%s;", DataSortovane)
Pm.FileTextWrite cesta , sRow, "mode:add;"
```

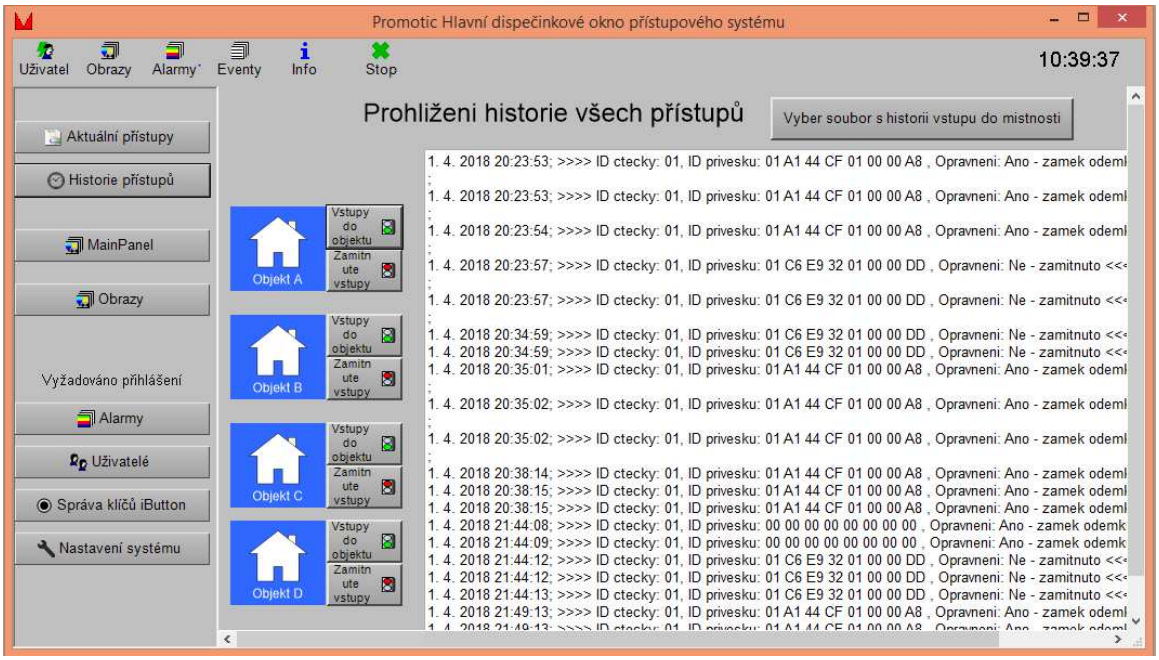
Obr. č. 35, Ukázka části WBSkriptu – ukládání historie přístupových interakcí
Zdroj.: vlastní

Jsou tak zaznamenávány všechny přístupové interakce a ukládány do souborů *pristupy_vsechny.txt, ct1_povolene.txt, ct2_povolene.txt, ct1_zamitnute.txt*, … a stejným způsobem jsou vytvářeny a plněny záznamy jednotlivé textové soubory pro archivaci přístupů do jednotlivých místností a archivaci přístupů všech registrovaných (a případně i neregistrovaných) iButton čipů.

7.2.3 Zobrazení záznamů o jednotlivých přístupových interakcích – dle místností a iButton čipů

Způsobem popsaným v předchozí podkapitole jsou automaticky vytvářeny textové soubory a do nich zaznamenávány přístupové interakce.
Obsah těchto souborů lze prohlížet i v aplikaci *AccessControl.pra*. Historie všech přístupových interakcí je zobrazována v systémovém okně v pravé části v panelu **Panel_pristupy**. Přímo k zobrazení historie přístupů pak slouží panel **Vypis_historie_pristupu.**

Tento panel je součástí obrazu **Vypis_historie_pristupu** a byl vytvořen jako předkonfigurovaný objekt PmPanel s editovatelným prohlížečem textu. Takto vytvořený objekt již obsahuje tlačítko ***ButtLoad*** (prvek typu PmiButton) po jehož stačení se automaticky vyvolá modální okno obsahující odkazy na textové soubory uložené v adresáři ***Data***, pokud jsou textové soubory přístupné pro čtení a adresář Data se nachází v domovském adresáři aplikace. Dále je součástí předkonfigurovaného objektu okno ***Viewer*** (objekt typu PmiWEdit), ve kterém lze zobrazovat obsah vybraných textových souborů.
Výše popsaný předkonfigurovaný objekt sloužící jako prohlížečka textových souborů byl upraven a doplněn o další tlačítka odkazující přímo na archiv přístupu pro určitou místnost, tak jak je patrné z obrázku č. 36.



Obr. č. 36, Promotic okno **Vypis_historie_pristupu** pro prohlížení historie povolených a zamítnutých vstupů do jednotlivých objektů (místností).
Zdroj.: vlastní

7.2 Nastavení přístupových práv v PC aplikaci

Aplikace ***AccessControl.pra*** dále disponuje funkcemi pro správu uživatel a nastavení přístupových práv. Informace o uživatelích jsou zaznamenávány pouze v aplikaci, zatímco přístupová práva pro jednotlivé iButton čipy jsou ukládána i do EEPROM paměti hlavního mikrokontroleru.
Funkce pro zadávání nových iButton čipů, mazání iButton čipů, přiřazování uživatelů a nastavování přístupových práv jsou dostupné prostřednictvím panelu **Sprava_uzivatel,** jenž je nakonfigurován k zadávání osobních údajů a přístupových práv pro 6 uživatelů.

Obr. č. 37, Promotic okno s panelem **Sprava_uzivatel** pro aktivaci iButton čipů, přiřazení uživatelů a nastavené přístupových práv
Zdroj.: vlastní

Nový uživatel je založen vyplněním jeho jméno a příjmení, dále je uživateli přidělen iButton čip, je mu nastavena právo pro přístup do aplikace AccesControl (pouze administrátorovi) a zadána práva pro přístup do jednotlivých místností. Přidělení iButton čipu se provede přiložením čipu ke čtečce a následným kliknutím na tlačítko „Přidělit čip“

Jednotlivé buňky sloužící k zadávání informací o uživatelích jsou grafickými objekty typu PmiWEdit a každá buňka má nastavenu datovou vazbu na příslušnou proměnou definovanou v objektu **Data_CipyUzivatele** (objekt typu PmData). Proměnné sloužící k dočasnému uchování dat o uživatelích mají nastaveno datové rozšíření typu ***WriteAction*** sloužící k vyvolávaní události během zápisu do hodnoty. Při změně hodnot proměnných je vyvolána událost **OnItemAfterWrite** definovaná na záložce +***Události*** v objektu **Data_CipyUzivatele**. Tato událost zajistí spuštění WBSkriptu, který provede odeslání nastavených hodnot do EEPROM paměti hlavního mikrokontroleru a zápis nastavených/změněných dat do souboru **Nastaveni_CipyUzivatele.ini** WBSkriptem je rovněž řešeno načtení dat ze souboru **Nastaveni_CipyUzivatele.ini** o uživatelích při spuštění aplikace.

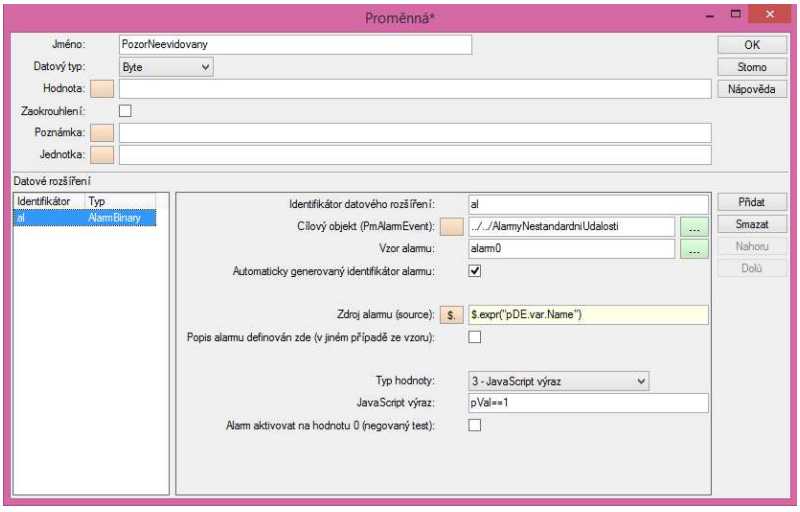
7.3 Zaznamenávání alarmních událostí a eventů

Alarmy a eventy v systému PROMOTIC slouží pro zachytávání a správu technologických dějů. **Alarm je** zachycený děj (např. porucha ventilu), který vznikne, zanikne a je potřeba aby ho uživatel vzal na vědomí tzv. kvitací. **Event je** méně závažný typ děje, který pouze vznikne a je potřeba ho zaznamenat. Protože oba pojmy jsou velmi podobné, řeší se tato oblast v systému PROMOTIC společně pomocí objektu PmAlarmEvent. [30]

V aplikace AccessControl je alarmní událost vyvolána v případě pokusu o vstup s neregistrovaným iButton čipem nebo v případě zkratu snímače čtečky po dobu delší než 30 sekund. Událost typu Event je pak zaznamenána v případě zamítnutého požadavku na vstup s registrovaným čipem (pokus o vstup do místnosti pro niž není čipu přiděleno právo ke vstupu), nebo při zkratu snímače čtečky po dobu delší než 5 sekund.

V aplikaci AcessControl byla vytvořena obrazová okna **AlarmNeevidovanyCip, AlarmZkrat, EventZamitnutyVstup, EventZkrat**. Dále byly v objektu **iButtonData** vytvořeny proměnné **AlarmNeevidovany, AlarmZkrat, PozorZamitnuty** a **PozorZkrat** a těchto proměnným bylo nastaveno datové rozšíření typu AlarmBinari (alarmování Boolean hodnoty).

Vytvořené alarmy a eventy fungují tak, že data přicházející z hlavního mikrokontroleru jsou v okamžiku přijetí (vyvolaní události ***onDataReceive*** v objektu ***iButtonMsg***) vyhodnocena a pokud je zjištěna nestandardní událost /např. adresa iButtonu není v aplikaci evidována/ je do příslušné proměnné /v tomto případě do **AlarmNeevidovany**/ uložena jednička. Díky nastavenému datovému rozšíření AlarmBinary a správně nastavené cesty k alarmu, tak jak je uvedeno na obrázku č. 19, je vyvolána alarmní událost. Stejným způsobem je řešeno spuštění alarmní události v případě příjmu řetězce obsahujícího pouze signál úrovně log. 1 (zkrat) a zaznamenávání eventů. Alarmy a eventy jsou zaznamenávány do databáze *dBase a* lze prohlížet v příslušných *obrazových oknech v aplikaci AccessControl.pra*



Obr. č. 38, Promotic okno vlastností proměnné PozorNeevidovany – datové rozšíření AlarmBinari a nastavení cesty na vytvořený alarm **AlarmNeevidovanyCip**.
Zdroj.: vlastní

8. vytvoření výukového materiálu

Poté co byl celý systém sestaven, odzkoušen a odladěn, byla realizována poslední část diplomové práce, tedy vytvoření výukového materiálu ve formě multimediální prezentace.

Cílem vytvořeného materiálu je shrnout nejdůležitější informace z teoretické části diplomové práce, tedy z kapitol 1. a 2., stručně čtenáře informovat o některých podstatných souvislostech obsažených v dalších kapitolách, jako je návrh a realizace hardwarové části s objasněním komunikace mezi jednotlivými moduly kontaktního přístupového systému prostřednictvím sběrnic a popis činnosti programů – firmwaru pro mikrokontrolery.

Naopak ve výrazně větší míře se výukový materiál věnuje aplikaci **AccessControl.pra**. V diplomové práci je obslužné aplikaci věnována 7. kapitola, ta je ovšem pojata jako seznámení s aplikací AccesControl po stránce poskytovaných funkcí a technologického řešení těchto funkcí, nevěnuje se však popisu pracovnímu, tedy využití aplikace z pohledu dispečera přístupového systému s právy běžného uživatele a z pohledu administrátora přístupového systému.

Realizovaný výukový materiál v podobě prezentace bude sloužit jako uživatelský manuál, který seznámí studenty a případné další uživatele s realizovaným systémem, jeho zprovozněním, obsluhou a nastavením. Tento výukový je k dispozici na CD, jež je součástí přílohové části diplomové práce.

Před obhajou diplomové práce budou materiály rovněž k dispozici na webové stránce autora na www.ibutton.tode.cz

Obr. č. 39: Realizovaný prototyp kontaktního přístupového systému se 4 čtečkami čipů Maxim iButton
Zdroj.: vlastní

9. Závěr

Cílem práce bylo navrhnout a realizovat přístupový systém se 4 čtečkami čipů Maxim iButton.

Nejprve byla provedena rešerše kontaktních přístupových systémů a následně srovnání jednotlivých technologií využívajících kontaktní identifikační prvek.
Další část byla věnována návrhu koncepce přístupového systému a definici požadavků pro volbu vhodných mikrokontrolerů a realizaci sběrnic. Následně práce seznamuje s parametry mikrokontrolerů PIC18LF46K22 a PIC16F18326 a dále konkretizuje všechny komponenty navrženého přístupového systému.

Takto navržený systém byl prakticky zrealizován. Jako hlavní modul byla využita vývojová deska s procesorem PIC18LF46K22, zapojení čtyř mikrokontrolerů čteček bylo realizováno v nepájivém poli. Pro vývoj firmwaru pro mikrokontrolery bylo využito vývojové prostředí od společnosti Microchip, MP LAB-X IDE. Systém byl vyvíjen v několika krocích a v souvislosti s tím byl software mnohokrát upravován, a to zejména software pro hlavní mikrokontroler. Úpravy systému i zdrojového kódu jsou v práci uvedeny a popsány.

Pro přístupový systém byla dále vytvořena obslužná aplikace AccessControl.pra ve SCADA prostředí Promotic. Realizace této aplikace je rovněž v práci zdokumentována.

Poslední část diplomové práce představuje realizovaný systém ve formě multimediální prezentace, která je zároveň uživatelským manuálem a popisuje ovládání počítačové aplikace AccessControl.pra

10. Použitá literatura

[1] LUKÁŠ, Luděk. *Bezpečnostní technologie, systémy a management*. Zlín: Radim Bačuvčík - VeRBuM, 2011. ISBN 978-80-87500-05-7.

[2] KŘEČEK, Stanislav. *Příručka zabezpečovací techniky*. 3. Vyd. 2006. ISBN 80-902938-2-4.

[3] FERRAIOLO, David., D. Richard. KUHN a Ramaswamy. CHANDRAMOULI. Role-based access control. Boston: Artech House, c2003. ISBN 1580533701.

[4] KERISYS. Why access control. Kerisys [online]. 2017 [cit. 2017-12-01]. Dostupné z: https://www.kerisys.com/pages/products/why-access-control/

[5] FERRARI, Elena. *Access control in data management systems*. San Rafael, calif.: Morgan & Claypool, 2010. ISBN 1608453758.

[6] DOSTÁLEK, Libor, Marta VOHNOUTOVÁ a Miroslav KNOTEK. *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*. 2., aktualiz. vyd. Brno: Computer Press, 2009. ISBN 978-80-251-2619-6.

[7] BURDA, Karel a Ivo STRAŠIL. *Zabezpečovací systémy.* Brno: VUT, 2011, ISBN 978-80-214-4441-6.

[8] OLMR, Vít. Co je to iButton. *Vývoj HW* [online]. 2006 [cit. 2017-12-01]. Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/co-je-to-ibutton.html

[9] ATERM.CZ. Čtečky čipů Dallas. *Aterm.cz* [online]. [cit. 2017-12-01]. Dostupné z: http://www.aterm.cz/Aterm.htm#2402

[10] AMPERTECH. přístupový terminál CX105. *Aterm.cz* 2017 [cit. 2017-12-01]. Dostupné z: http://www.ampertech.cz/pristupove-systemy-rfid-dallas/587-cipovy-terminal-cx105.html

[11] TOMST. *Přístupový systém garant TDL*. Praha: Atis group, 2002, ISBN neuvedeno, s. 4.

[12] ŘEHOŘOVÁ, Kateřina. Docházkové systémy a evidence pracovní docházky *Mineralfit.cz* [online]. 2006 [cit. 2018-01-25]. Dostupné z: https://mineralfit.cz/clanek/dochazkove-systemy-a- evidence-pracovni-dochazky

[13] IBUTTONLINK.COM. IButton. IButtonlink.com [online]. 2018 [cit. 2018-01-25]. Dostupné z: https://www.ibuttonlink.com/collections/temperature-logging-ibuttons

[14] BENÁČEK, Martin. Možnosti identifikace a monitorování pohybu zboží a osob. Brno, 2007. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Pavel Houška, Ph.D

[15] Čárové kódy, RFID, magnetické karty— Transkript prezentace. *Slide player* [online]. [cit. 2017-12-01]. Dostupné z: http://slideplayer.cz/slide/3203247/

[16] DS1990R, maxim integrated. online]. 2018 [cit. 2018-01-25]. Dostupné z: https://www.maximintegrated.com/en/products/digital/memory-products/DS1990R.html

[17] *ecom.cz | ecom.cz* [online]. Copyright © [cit. 28.06.2018]. Dostupné z: https://www.ecom.cz/edited/soubory-editoru/File/novinky2015pic2.pdf

[18] *Distrelec Germany | Best Online Shop for Electronics* [online]. Copyright © 2018 Distrelec GmbH. All rights reserved. [cit. 28.06.2018]. Dostupné z: https://www.distrelec.de/en/pic18f4xk22-development-board-microchip-dm164134/p/30012914

[19] 8-bit Low Power MCUs | Microchip Technology . *Home* | *Microchip Technology* [online]. Copyright ©Copyright 1998 [cit. 29.06.2018]. Dostupné z: https://www.microchip.com/design-centers/lowpower/products/8-bit-low-power-mcus

[20] Low Power MCU and MPUs | Microchip Technology . *Home* | *Microchip Technology* [online]. Dostupné z: https://www.microchip.com/design-centers/lowpower

[21] *Analog, Embedded Processing, Semiconductor Company, Texas Instruments - TI.com* [online]. Dostupné z: http://www.ti.com/product/sn75176a

[22] NETÁHLO, Tomáš. *PALUBNÍ MULTIFUNKČNÍ JEDNOTKA PRO MOTOCYKLY*. Brno, 2012. Diplomová práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/10637/Diplomova_prace.pdf?sequence=2

[23] Interfacing The Serial / RS-232 Port. *Beyond Logic* [online]. Dostupné z: https://retired.beyondlogic.org/serial/serial.htm

[24] HENZL, Václav. *PŘEVODNÍK USB/RS-485*. Brno, 200. Bakalářská práce. VUT Brno, Fakulta elektrotechniky a komunikačních technologií. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=8768

[25] Převodníky RS 485 / www.papuch.com [online]. Copyright © [cit. 21.06.2018]. Dostupné z: https://www.papouch.com/cz/shop/products/prevodniky/rs485/

[26] Převodníky USB//UART, slavino.sk [online]. Dostupné z: https://slavino.sk/elektro/navody/prevodniky-usb---uart.html

[27] PIC16F18323 - Microcontrollers and Processors - Microcontrollers and Processors. *Home* | *Microchip Technology* [online]. Copyright ©Copyright 1998 [cit. 28.06.2018]. Dostupné z: https://www.microchip.com/wwwproducts/en/PIC16F18323

[28] PIC18F4XK22 Development Board. *Home* | *Microchip Technology* [online]. 1998 Dostupné :http://www.microchip.com/DevelopmentTools/ProductDetails/dm164134

[29] MPLAB IDE | Microchip Technology . *Home* | *Microchip Technology* [online]. Copyright ©Copyright 1998 [cit. 28.06.2018]. Dostupné z: http://www.microchip.com/mplab

[30] What tools do I need to develop PIC® - Developer Help. *Home - Developer Help* [online]. Copyright © 2018 Microchip Technology, Inc. [cit. 27.06.2018]. Dostupné z: http://microchipdeveloper.com/tools:what-do-i-need

[31] MPLAB XC8 Compiler PRO Dongle License. *Home* | *Microchip Technology* [online]. Copyright ©Copyright 1998 [cit. 29.06.2018]. Dostupné z: https://www.microchip.com/DevelopmentTools/ProductDetails/SW006021-DGL

[33] Promotic Tutorial, [online]. Copyright © MICROSYS, spol. s r. o. [cit. 27.06.2018]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Tutorial/TutorStart.htm

[34] PROMOTIC - Přehled ke stažení. [online]. Copyright © MICROSYS, spol. s r. o. [cit. 27.06.2018]. Dostupné z: https://www.promotic.eu/cz/promotic/download/download.htm

Seznam příloh

PŘÍLOHA A: FOTOGRAFIE REALIZOVANÉHO PŘÍSTUPOVÉHO SYSTÉMU

PŘÍLOHA B: ZDROJOVÝ KÓD V SOUBORECH PRO HLAVNÍ MIKROKONTROLER

iButton_master.c,

iButton_master_iButtons.c,

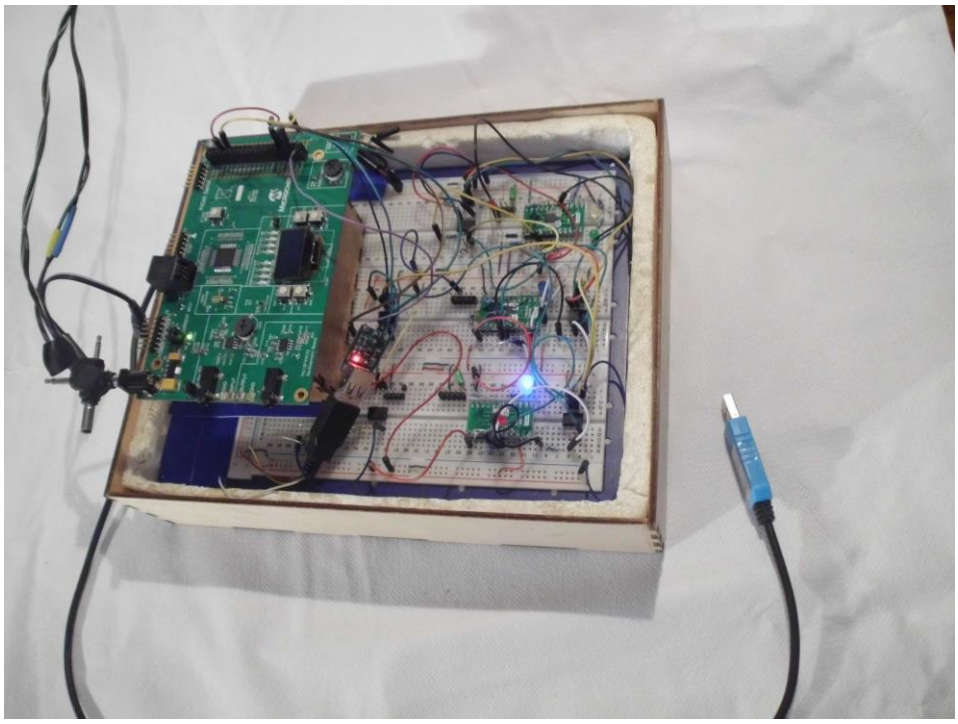
iButton_master.h

iButton_master_retezce.h

PŘÍLOHA C: ZDROJOVÝ KÓD V SOUBORECH PRO MODULY ČTEČEK

Součástí DP je CD.
Adresářová struktura přiloženého CD:

PŘÍLOHA A: FOTOGRAFIE REALIZOVANÉHO PŘÍSTUPOVÉHO SYSTÉMU



Na 1. fotografii je realizace zapojení v nepájivém poli

Na 2. fotografii výsledný prototyp přístupového systému uložený ve dřevěném boxu.

Příloha A: *Schéma zapojení*

PŘÍLOHA B

Zdrojové soubory pro hlavní mikrokontroler

1) iButton_master.h
2) iButton_master_retezce.h
3) iButton_master_iButtons.c,
4) iButton_master.c

1) Zdrojový soubor iButton_master.h

```
//###########################################################################################
//###########################################################################################
/*
    iButton_master.h

    iButton - MASTER            PIC18LF46K22


*/
//###########################################################################################
#
//###########################################################################################
#
#include <stdint.h>


//#########################################################################
##################

//LED:
#define LEDY             LATC4
#define LEDG             LATC3
#define LEDR             LATC2

// UART1 - PC:
#define UART1_TX         PORTCbits.RC6
#define UART1_RX         PORTCbits.RC7

// UART2 - RS485:
#define UART2_TX         PORTBbits.RD6
#define UART2_RX         PORTBbits.RD7
#define Flow_control     LATD2            //H=Tx, L=Rx


//#########################################################################
##################

#define off       1
#define on        0

#define input   1
#define output  0

#define RS485   0
#define PC      1

#define  YES    1

#define found_match 1

#define transmit  1
#define receive   0

//#########################################################################
##################
```

```
char  Buffer_Rx_UART1[100];
char  Buffer_Rx_UART2[100];

bit         Flag_Rx_UART1;           //prijata nova data (prikazy) z PC od
Rx UARTU
bit         Flag_Rx_UART2;           //prijata nova data (prikazy) z RS485

bit     Flag_temp;              //Univerzalni flag
bit     Flag_match;             //Nastavi se, kdyz je v pameti nalezen
shodny kod iButton
bit     Flag_send_answer;    //Po prijmu kodu od Slave posila zpaet
potvrzeni o autorizaci ANO/NE (na Slave se rozsviti LED R/G)

char    Registr_temp;
char    ID_Slave;

char    MSB, LSB;
```

2) Zdrojový soubor iButton_master_retezce.h

```
//#############################################################################################
//#############################################################################
/*
    iButton_master_retezce.h

    iButton - MASTER            PIC18LF46K22


*/
//#############################################################################################
#
//#############################################################################################
#
#include <stdint.h>


//#################################TX STRINGs:
        #define    S_Code_Slave           "ReaderNO:"

        #define    S_Code_iButton          ", iButtonID: "

        #define    S_AccessRights           ", AccessRights: "

        #define    S_YES               "YES - lock unlock"

        #define    S_NO                "NO - access danied"

        #define    S_unknown_string       " >>>> unknown string <<<<"

//#############################################################################################
//#############################################################################################
```

3) Zdrojový soubor iButton_master_iButtons.h

```
//#############################################################################################
//#############################################################################################
/*
    iButton_master_iButtons.h

    iButton - MASTER            PIC18LF46K22


*/
//#############################################################################################
#
//#############################################################################################
#

#include <stdint.h>

#define pravo_1 0x01
#define pravo_2 0x02
#define pravo_3 0x04
#define pravo_4 0x08


//############################## Kody iButton s opravnenim - zadano v HEX

// const char __section("ROMDATA") iButtons_code_array[];

const char iButtons_code_array[] =
                        {
                            0x01,0xA1,0x44,0xCF,0x01,0x00,0x00,0xA8,
//ID iButton 1 (zleva Family, ID, CRC)
                            0x01,0xA2,0xB2,0xD6,0x01,0x00,0x60,0x60,
//ID iButton 2
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 3
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 4
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 5
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 6
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 7
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 8
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
//ID iButton 9
                            0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00
//ID iButton 10
                        };
```

4) Zdrojový soubor iButton_master.c

```
/*
    iButton_master.c

    iButton - MASTER            PIC18LF46K22


    IRC8 MHz
                1. UART (RS485)
                2. UART to PC
                1 x LED

               18.6.2018

*/
//#############################################################################################
#
//#############################################################################################
#
#include <htc.h>
#include "iButton_master.h"
#include "iButton_master_retezce.h"
#include "iButton_master_iButtons.h"
#include <xc.h>
//#############################################################################################
// CONFIG1H
#pragma config FOSC = INTIO67   // Oscillator Selection bits (Internal oscillator block)
#pragma config PLLCFG = OFF     // 4X PLL Enable (Oscillator used directly)
#pragma config PRICLKEN = ON    // Primary clock enable bit (Primary clock enabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover bit

// CONFIG2L
#pragma config PWRTEN = OFF     // Power-up Timer Enable bit (Power up timer disabled)
#pragma config BOREN = SBORDIS  // Brown-out Reset Enable bits
#pragma config BORV = 190       // Brown Out Reset Voltage bits (VBOR set to 1.90 V nominal)

// CONFIG2H
#pragma config WDTEN = ON      // Watchdog Timer Enable bits, SWDTEN has no effect.)
#pragma config WDTPS = 32768    // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H
#pragma config CCP2MX = PORTC1  // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF     // PORTB A/D Enable bit (PORTB<5:0> pins are configured as digital I/O
on Reset)
#pragma config CCP3MX = PORTB5  // P3A/CCP3 Mux bit (P3A/CCP3 input/output is multiplexed with RB5)
#pragma config HFOFST = OFF     // HFINTOSC Fast Start-up (HFINTOSC output and ready status are
delayed by the oscillator stable status)
#pragma config T3CMX = PORTC0   // Timer3 Clock input mux bit (T3CKI is on RC0)
#pragma config P2BMX = PORTB5   // ECCP2 B output mux bit (P2B is on RB5)
#pragma config MCLRE = INTMCLR  // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

// CONFIG4L
```

```
#pragma config STVREN = OFF     // Stack Full/Underflow Reset Enable bit (Stack full/underflow will not cause Reset)
#pragma config LVP = OFF        // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Instruction set extension)

// CONFIG5L
#pragma config CP0 = OFF        // Code Protection Block 0 (Block 0 (000800-003FFFh) not code-protected)
#pragma config CP1 = OFF        // Code Protection Block 1 (Block 1 (004000-007FFFh) not code-protected)
#pragma config CP2 = OFF        // Code Protection Block 2 (Block 2 (008000-00BFFFh) not code-protected)
#pragma config CP3 = OFF        // Code Protection Block 3 (00C000-00FFFFh) not code-protected)

// CONFIG5H
#pragma config CPB = OFF        // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)Z#pragma config CPD = OFF        // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection Block 0 (Block 0 (000800-003FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection Block 1 (Block 1 (004000-007FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection Block 2 (Block 2 (008000-00BFFFh) not write-protected)
#pragma config WRT3 = OFF       // Write Protection Block 3 (Block 3 (00C000-00FFFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection bit
#pragma config WRTB = OFF       // Boot Block Write Protection bit #pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF      // Table Read Protection Block
#pragma config EBTR1 = OFF      // Table Read Protection Block 1
#pragma config EBTR2 = OFF      // Table Read Protection Block 2
#pragma config EBTR3 = OFF      // Table Read Protection Block 3

// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot Block (000000-0007FFh) not protected from table reads executed in other blocks)

//#########################################################################################################
#
void Delay(char cas);

//#########################################################################################################
#
__interrupt()

void preruseni (void)
  {

      unsigned char ukazatel;
      unsigned short timer_Rx;

//------------------------- Preruseni od UARTU 1 (PC)
            if(RC1IF)
                  {
            ukazatel=0;
            timer_Rx=0;

            Flag_Rx_UART1=1;
```

```
                        do{
                                        if(RC1IF)
                                                {
                                                        Buffer_Rx_UART1[ukazatel]=RCREG1;
                                                        if(ukazatel<101) ++ukazatel;
                                                        timer_Rx=0;
                                                }

                                        }while(++timer_Rx<250);

            while(RC1IF) { RC1IF=0; Registr_temp=RCREG1;}

                    }

//------------------------- Preruseni od UARTU 2 (RS485)
             if(RC2IF)
                    {
                        ukazatel=0;
                        timer_Rx=0;

                        Flag_Rx_UART2=1;

                        do{
                                        if(RC2IF)
                                                {
                                                        Buffer_Rx_UART2[ukazatel]=RCREG2;
                                                        if(ukazatel<101) ++ukazatel;
                                                        timer_Rx=0;
                                                }

                                        }while(++timer_Rx<250);

            while(RC2IF) { RC2IF=0; Registr_temp=RCREG1;}

                    }

 }
//#############################################################################################
void nastaveni_procesoru()
{

   OSCCON  =  0B01100010;     //8MHz
   OSCCON2 =  0B00000111;
//
          PORTA          =  0xFF;
          PORTB          =  0xFF;
          PORTC          =  0xFF;
//
   TRISA          =  0B00000000;
   TRISB   =  0B10000000;
   TRISC          =  0B10000000;

//
   WDTCON  =  0B00011001;

   INTCON  =  0x00;
   INTCON2 =  0x00;
```

```
    INTCON3 =  0x00;
    PIR1    =  0x00;
    PIR2    =  0x00;
    PIR3    =  0x00;
    PIR4    =  0x00;
    PIR5    =  0x00;

            PIE1    = 0x00;
            PIE2    = 0x00;
            PIE3    = 0x00;
            PIE4    = 0x00;
            PIE5    = 0x00;

    PMD0    = 0xFF;
    PMD1    = 0xFF;
    PMD2    = 0xFF;



    ANSELA  =  0x00;
    ANSELB  =  0x00;
    ANSELC  =  0x00;

    WPUB    =  0;

    FVRCON  =  0x00;

    ADCON0  =  0x00;
    ADCON1  =  0x00;

    CM1CON0 =  0x00;
    CM2CON0 =  0x00;

            CCP1CON = 0;
    CCP2CON = 0;
    CCP5CON = 0;

            SSP1CON1= 0;
            SSP1CON3= 0;
    SSP2CON1= 0;
    SSP2CON3= 0;


            TMR1ON          = 0;
            TMR2ON  = 0;
    T2CON   = 0;

    ECCP1AS = 0x00;
    ECCP1AS = 0x00;

    CTMUCONH = 0x00;
    SLRCON   = 0x00;



}
```

```
//#############################################################################################
#################
void Nastaveni_UART1()
{
//---------- Nastaveni USART1 - PC

        TXSTA1          = 0B00100100;     //OSC8MHz / 9600B
        RCSTA1          = 0B10000000;
        BAUDCON1 = 0B00001000;
        SPBRG1          = 207;
        SPBRGH1         = 0;
        while(RC1IF) { RC1IF=0; NOP(); Registr_temp=RCREG1;}
  SPEN1    = 1;
  CREN1    = 1;
  RC1IE    = 1;
  UART1MD  = 0;
//Povoleni preruseni
  GIE_GIEH  = 1;
  PEIE_GIEL = 1;
}
//#############################################################################################
void Nastaveni_UART2()
{

//---------- Nastaveni USART2 - RS485

        TXSTA2          = 0B00100100;     //OSC8MHz / 9600B
        RCSTA2          = 0B10000000;
        BAUDCON2 = 0B00001000;
        SPBRG2          = 207;
        SPBRGH2         = 0;
        while(RC2IF) { RC2IF=0; NOP(); Registr_temp=RCREG;}
  SPEN2    = 1;
  CREN2    = 1;
  RC2IE    = 1;
  UART2MD  = 0;
//Povoleni preruseni
  GIE_GIEH  = 1;
  PEIE_GIEL = 1;
}
//#############################################################################################

void Delay(char cas)
{
  unsigned int  e=0;

  do {  e++;

                {
                        char k;
                        for (k=cas;k>1;k--)         continue;
                }
        } while (e<589);
}
//#############################################################################################
void PosliZnak(char UARTx, char znak)
{
  if(UARTx)
```

```
   {
      //-------- PC
      while(!TX1IF);
      TXREG1 = znak; NOP();NOP();NOP();NOP();NOP();
      while(!TX1IF);
   }
   else
   {
      //-------- RS485
      while(!TX2IF);
      TXREG2 = znak; NOP();NOP();NOP();NOP();NOP();
      while(!TX2IF);
   }
}
//*****************************************************************************************
*****************
void PosliText (char UARTx, char *s) // posila retezce na UART
{
          do {
      char znak = *s++;
      if (!znak) {break; }
      if(UARTx) { TX1REG = znak; } else { TX2REG = znak; }
      NOP(); NOP(); NOP();
      if(UARTx) { while(!TX1IF) {NOP(); }; } else while(!TX2IF) {NOP(); };
     } while (1);
}
//#########################################################################################
void prevod_HEX_Ascii(char vstup_hex)
{
   MSB=vstup_hex>>4; LSB=vstup_hex&0x0F;
   if(MSB>9) { MSB=MSB+55; } else { MSB=MSB+48; }
   if(LSB>9) { LSB=LSB+55; } else { LSB=LSB+48; }
}
//#########################################################################################
void Vymaz_Buffer_Rx_UART1()
{
          for(char i=0;i<101;i++) Buffer_Rx_UART1[i]=0xFF;
}
//#########################################################################################
void Vymaz_Buffer_Rx_UART2()
{
          for(char i=0;i<101;i++) Buffer_Rx_UART2[i]=0xFF;
}
//#########################################################################################
void Restart_UART1()
{
          Flag_Rx_UART1=0;
          Nastaveni_UART1();
          Vymaz_Buffer_Rx_UART1();
}
//#########################################################################################
void Restart_UART2()
{
          Flag_Rx_UART2=0;
          Nastaveni_UART2();
          Vymaz_Buffer_Rx_UART2();
}
void Vyhodnot_vystup(char Flag_opravneni)
```

```
{
  if(Flag_opravneni==ano)
  {
    LEDG=on;                    //opravneni OK, sepnuti RELE (zelena LED)

 //-------- Nasleduje odeslani informace do PC: --------------------
    PosliText(PC,(char*)" >>>> ");     //uvodni znaky

    PosliText(PC,(char*)S_Kod_Slave);    //kod ctecky
    prevod_HEX_Ascii(Buffer_Rx_UART2[1]);
    PosliZnak(PC,MSB); PosliZnak(PC,LSB); //... odeslan na PC v ASCII

    PosliText(PC,(char*)S_Kod_iButton);  //kod iButton

    char pozice=2;
    do{                         //odesilani kodu iButton v ASCII formatu
      prevod_HEX_Ascii(Buffer_Rx_UART2[pozice]);
      PosliZnak(PC,MSB); PosliZnak(PC,LSB); PosliZnak(PC,' ');
    }while(++pozice<10);

    PosliText(PC,(char*)S_Opravneni);    //Opraveni ANO
    PosliText(PC,(char*)S_Ano);

    PosliText(PC,(char*)" <<<< ");     //ukoncovaci znaky
  }
  else
  {
    LEDR=on;                    //opravneni NOK, nezapnuti RELE (zelena LED)
  //-------- Nasleduje odeslani informace do PC: --------------------
    PosliText(PC,(char*)" >>>> ");     //uvodni znaky

    PosliText(PC,(char*)S_Kod_Slave);    //kod ctecky ...
    prevod_HEX_Ascii(Buffer_Rx_UART2[1]);
    PosliZnak(PC,MSB); PosliZnak(PC,LSB); //... odeslan na PC v ASCII


    PosliText(PC,(char*)S_Kod_iButton);  //kod iButton

    char pozice=2;
    do{                         //odesilani kodu iButton v ASCII formatu
      prevod_HEX_Ascii(Buffer_Rx_UART2[pozice]);
      PosliZnak(PC,MSB); PosliZnak(PC,LSB); PosliZnak(PC,' ');
    }while(++pozice<10);

    PosliText(PC,(char*)S_Opravneni);    //Opraveni NE
    PosliText(PC,(char*)S_Ne);

    PosliText(PC,(char*)" <<<< ");     //ukoncovaci znaky
  }
}
//#############################################################################################
bit Porovnej_kody_iButton()
{
  char Buff_pointer=2;
  char Page_pointer=0;

    do{
```

```
        if(Buffer_Rx_UART2[Buff_pointer]==iButtons_code_array[(Buff_pointer-2)+Page_pointer])
//Prohledava pole a porovnava s prijatym kodem iButton
            {
              if(Buff_pointer==9) { return 1; } else { ++Buff_pointer; }
            }
            else
            {
              Page_pointer=Page_pointer+8; Buff_pointer=2;
            }

        }while(Page_pointer<=72);

  return 0;
}
//#############################################################################################
void Vyhodnot_data_od_UART1()
{

}
//#############################################################################################
void Vyhodnot_data_od_UART2()
{
  Flag_shoda=0;

  if(Buffer_Rx_UART2[0]=='S' && Buffer_Rx_UART2[10]=='E'  )        //Pokud vysila Slave a je v poradku
startovaci a ukoncovali znak najdi shodu iButton a sepni rele
  {
    ID_Slave=Buffer_Rx_UART2[1];                                   //Tady je ulozeno cislo ctecky
    if(Porovnej_kody_iButton()==nalezena_shoda) Flag_shoda=1;      //Porovna se prijate ID iButton s temi,
ulozenymi v pameti
    Vyhodnot_vystup(Flag_shoda);
    Flag_odeslat_odpoved=1;
  }
}
//#############################################################################################
void Odeslat_odpoved(char ID_Slave, char Flag_shoda)
{
  Rizeni_toku=vysilani;
  Delay(10);

  PosliZnak(RS485,'M');                                //Startovaci znak od Master je 'M'
  PosliZnak(RS485,ID_Slave);                             //Adresat
  if(Flag_shoda)
  {
    PosliZnak(RS485,0xF1);                             //Prikaz (0xF0=rozsvit cervenou 0xF1=rozsvit zelenou
LED)
  }
  else
  {
    PosliZnak(RS485,0xF0);                             //Prikaz (0xF0=rozsvit cervenou 0xF1=rozsvit zelenou
LED)
  }
  PosliZnak(RS485,'E');                                //Ukoncovaci znak

  Delay(1);
  Rizeni_toku=prijem;
}
//#############################################################################################
```

```
void main (void)
{
        nastaveni_procesoru();
   Rizeni_toku=prijem;

//---------------------------------------- NASTAVENI ---------------------------

        Nastaveni_UART1();
   Nastaveni_UART2();

        Restart_UART1();
   Restart_UART2();

        Vymaz_Buffer_Rx_UART1();
   Vymaz_Buffer_Rx_UART2();

   GIE=1;          //Globalni povoleni preruseni (pouzivame od Uartu))
   PEIE=1;

   LEDG=off;
   LEDR=off;
   LEDY=off;


 //---------------------------------------- HLAVNI PROVOZNI SMYCKA ---------------------------------
        do{
      Flag_odeslat_odpoved=0;
      Rizeni_toku=prijem;
//------------ Zluta, stavova LED ------------------------------------------------------------------
      LEDY=!LEDY; //Stavova LED. Pokud je zarizeni pripraveno, blika.
//------------ Vypnuti ostatnich LED ---------------------------------------------------------------
      LEDG=off; LEDR=off;
//------------ Jen prodleva definujici rychlost blikani zlute LED ----------------------------------
      Delay(50);
//------------ Vyhodnoceni dat od  PC --------------------------------------------------------------
      if(Flag_Rx_UART1) { Vyhodnot_data_od_UART1(); Restart_UART1(); Flag_Rx_UART1=0; }
//------------ Vyhodnoceni dat od RS485 (iButton) --------------------------------------------------
      if(Flag_Rx_UART2) { LEDY=on; Vyhodnot_data_od_UART2(); Restart_UART2(); Flag_Rx_UART2=0;
}
//------------ Pokud byl prijem od Slave uspesny, je odeslano potvrzeni ----------------------------
//------------ Pak je jeste pridana delsi prodleva definujici svit cervene/zelene LED --------------
      if(Flag_odeslat_odpoved) { Odeslat_odpoved(ID_Slave,Flag_shoda); Flag_odeslat_odpoved=0;
Delay(100); }
//----------- Nulovani WDT -------------------------------------------------------------------------
                CLRWDT();
//---------- Konec hlavni smycky -------------------------------------------------------------------

        }while(1);


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$

}
```

PŘÍLOHA C

Zdrojové soubory pro mikrokontroler čtečky

1) iButton_slave.h
2) iButton_RETEZCE.h
3) iButton_slave.c

1) Zdrojový soubor iButton_slave.h

```
//#############################################################################################
//#############################################################################
/*
    iButton_slave..h

    iButton - SLAVE             PIC16F18323


*/
//#############################################################################################

#include <stdint.h>


//#############################################################################

//LED:
#define LEDB          LATC2
#define LEDG          LATA5
#define LEDR          LATA4

//1-Wire
#define Port_1Wire  RC0

// UART + rizeni RS485:
#define UART_TX         RC4
#define UART_RX         RC5
#define Rizeni_toku   LATC3       //H=Tx, L=Rx



//#############################################################################
#

#define off       1
#define on        0
#define input   1
#define output  0
#define prijem  0
#define vysilani 1

//#############################################################################
##################


char  Buffer_Rx_UART[101];    //Buufer pro prijem dat od Master po RS485
char    Buffer_iButton[10];


bit         Flag_Rx_UART;           //prijata nova data (prikazy) z RS485
```

```
od Rx UARTU (od Master
bit     Flag_temp;              //Univerzalni flag

char    Casovac_blokace;
bit     Flag_blokace;
```

2) Zdrojový soubor iButton_slave_retezce.h

```
//#############################################################################################
//#############################################################################
/*
    iButton_slave_retezce.h

    iButton - SLAVE             PIC16F18323

*/
//#############################################################################################
#
//#############################################################################################
#

#include <stdint.h>

//############################## TX STRINGs

//#define     iButton_num               "Cislo iButton: "       //Pomocny
string pri vyvoji, nepouziva se

//############################## Cislo ctecku iButton, ktere se vysila na
Master jako ID ctecky:

#define     ID_Slave                  0x02



//#############################################################################
################
```

3) Zdrojový soubor iButton_slave.c

```
//#########################################################################
################
/*

        iButton - SLAVE

            PIC 16F18323
        IRC8 MHz
            1 x 1-Wire vstup pro iButton
        1 x UART (RS485)
        1 x LED
            ....


            18.3.2017

*/
//#########################################################################
################
//#########################################################################
################
#include "htc.h"
#include "iButton_slave.h"
#include "iButton_slave_retezce.h"
#include <xc.h>
//#########################################################################
########################
// CONFIG1
#pragma config FEXTOSC = OFF    // FEXTOSC External Oscillator mode
Selection bits (Oscillator not enabled)
#pragma config RSTOSC = HFINT1  // Power-up default value for COSC bits
(HFINTOSC (1MHz))
#pragma config CLKOUTEN = OFF   // Clock Out Enable bit (CLKOUT function is
disabled; I/O or oscillator function on OSC2)
#pragma config CSWEN = ON       // Clock Switch Enable bit (Writing to NOSC
and NDIV is allowed)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enable (Fail-
Safe Clock Monitor is enabled)

// CONFIG2
#pragma config MCLRE = OFF      // Master Clear Enable bit (MCLR/VPP pin
function is digital input; MCLR internally disabled; Weak pull-up under
control of port pin's WPU control bit.)
#pragma config PWRTE = ON       // Power-up Timer Enable bit (PWRT enabled)
#pragma config WDTE = OFF       // Watchdog Timer Enable bits (WDT
disabled; SWDTEN is ignored)
#pragma config LPBOREN = OFF    // Low-power BOR enable bit (ULPBOR
disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable bits (Brown-out
Reset enabled, SBOREN bit ignored)
#pragma config BORV = LOW       // Brown-out Reset Voltage selection bit
(Brown-out voltage (Vbor) set to 2.45V)
```

```
#pragma config PPS1WAY = OFF    // PPSLOCK bit One-Way Set Enable bit (The
PPSLOCK bit can be set and cleared repeatedly (subject to the unlock
sequence))
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable
bit (Stack Overflow or Underflow will cause a Reset)
#pragma config DEBUG = OFF      // Debugger enable bit (Background debugger
disabled)

// CONFIG3
#pragma config WRT = ALL        // User NVM self-write protection bits
(0000h to 07FFh write protected, no addresses may be modified)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (High
Voltage on MCLR/VPP must be used for programming.)

// CONFIG4
#pragma config CP = ON          // User NVM Program Memory Code Protection
bit (User NVM code protection enabled)
#pragma config CPD = ON         // Data NVM Memory Code Protection bit
(Data NVM code protection enabled)

//#####################################################################
#########################
void Delay(char cas);
void Vymaz_Buffer_Rx_UART();
void Nastaveni_UARTU();
void Restart_UART();
//#####################################################################
#################################
//#####################################################################
#################################
//#####################################################################
#################################
__interrupt ()
void preruseni (void)
   {

//------------------------- Preruseni od UARTU Rx
          if(RCIF)
            {
                  unsigned char ukazatel=0;
                  unsigned short timer_Rx=0;
                  Flag_Rx_UART=1;

                  do{
                              if(RCIF)
                                    {
                                          Buffer_Rx_UART[ukazatel]=RC1REG;
                                          if(ukazatel<100) ++ukazatel;
                                          timer_Rx=0;
                                    }

                        }while(++timer_Rx<150);


                     while(RCIF) { RCIF=0; char X=RC1REG; }

               }

//------------------------- Preruseni od portu TMR2
```

```
            if(TMR2IF)
                  {
                        while(TMR2IF) { TMR2IF=0; }
                  }


//------------------------- Preruseni od portu RB

            if(IOCIF)
                  {

                        while(IOCIF) { IOCIF=0;}
                  }

 }
//#####################################################################
################################
//#####################################################################
################################
void nastaveni_procesoru()
{

    OSCCON1 =  0B01100000;      //IRC OSC 8MHz
    OSCCON2 =  0B01100000;
    OSCFRQ  =  0B00000100;
    OSCTUNE =  0;               //tuning register, default value

//
      PORTA       =  0B11111011;
      PORTC       =  0B11111111;
//
    TRISA   =  0B11001000;
    TRISC   =  0B11100001;

//
    WDTCON  =  0B00011001;
    INTCON  =  0x00;

    CLC1CON=0;
    CLC2CON=0;
    MDCARH=0;
    MDSRC=0;
    MDCON=0;

    PPSLOCK=0x55;
    PPSLOCK=0xAA;
    PPSLOCKED=0;

    RXPPS=0B00010101;
    TXPPS=0B00010100;
    RC4PPS=0B00010100;

    PPSLOCKED=1;

    PMD0 = 0;
    PMD1 = 0;
    PMD2 = 0xFF;
    PMD3 = 0xFF;
```

```
    PMD4 = 0B00000010;
    PMD5 = 0xFF;

    ANSELA  =  0x00;
    ANSELC  =  0x00;
    WPUA    =  0;
    WPUC    =  0B00100000;
    FVRCON  =  0x00;
    ADCON0  =  0x00;
    ADCON1  =  0x00;
    CM1CON0 =  0x00;
    CM2CON0 =  0x00;
      CCP1CON = 0;
      while(IOCIF) { IOCIF=0; }
      PIE1    = 0x00;
      PIE2    = 0x00;
      SSP1CON1= 0;
      SSP1CON3= 0;
      SSPEN = 0;
      TMR1ON      = 0;
      TMR2ON  = 0;
    T2CON   = 0;



}
//#####################################################################
##################################
void Nastaveni_UARTU()
{
      TX1STA       = 0B00100100;      //OSC8MHz / 9600B
      RC1STA       = 0B10000000;
      BAUD1CON = 0B00001000;
      SP1BRGL      = 207;
      SP1BRGH      = 0;
      while(RCIF) { RCIF=0; NOP(); }
      RCIE     = 1;
      PEIE     = 1;
    SPEN     = 1;
    CREN     = 1;
      GIE      = 1;
}
//#####################################################################
##################################
//#####################################################################
##################################
void Delay(char cas)
{
    unsigned int  e=0;

    do {  e++;

                {
                      char k;
                      for (k=cas;k>1;k--)     continue;
                }
          } while (e<589);
}
```

```
//*********************************************************************
*******************************
void Delay_10us(char casus)   //cas x 10uS
{
    while(--casus>0)
     {
        NOP(); NOP();NOP();NOP();NOP();NOP();NOP();NOP();NOP();NOP();NOP();
        NOP();NOP();NOP();NOP();
    }
}
//*********************************************************************
*******************************
void Delay_6us(void)
{
    NOP(); NOP(); NOP(); NOP(); NOP();
}
//*********************************************************************
*******************************
void Delay_9us(void)
{
    NOP(); NOP(); NOP(); NOP(); NOP(); NOP(); NOP();
}
//###################################################################
#################################
//###################################################################
#################################
void PosliZnak(char znak)
{
      while(!TXIF);
      TX1REG = znak; NOP();NOP();NOP();NOP();NOP();
      while(!TXIF);
}
//*********************************************************************
*******************************
void PosliText (char *s) // posila retezce na UART1
{
      do {
        char znak = *s++;
        if (!znak) {break; }
        TX1REG = znak;
        NOP(); NOP(); NOP();
        while(!TXIF) {NOP(); };
       } while (1);
}

//###################################################################
#################################
//###################################################################
#################################
void Vymaz_Buffer_Rx_UART()
{
      for(char i=0;i<100;i++) Buffer_Rx_UART[i]=0xFF;
}
//###################################################################
#################################
void Restart_UART()
{
      Flag_Rx_UART=0;
      Nastaveni_UARTU();
```

```
      Vymaz_Buffer_Rx_UART();
}
//#########################################################################
################################
//#########################################################################
################################
void SetPort_1Wire(char set_port)
{
if(set_port)
    {
        TRISC=TRISC|0x01;
    }
else
    {
        TRISC=TRISC&0xFE;
    }
}
//#########################################################################
################################
bit  iButton_Reset(void)
{
    Port_1Wire=0;
    SetPort_1Wire(output);
    Delay_10us(48);
    SetPort_1Wire(input);
    Delay_10us(7);

    if(Port_1Wire) { Delay_10us(45); return 0; }

    Delay_10us(45);
    return 1;

}
//#########################################################################
################################
bit iButton_ReadBit(void)
{
    char Dbit = 0;                                  // Dallas bit result

      Port_1Wire=0;
    SetPort_1Wire(output);
    Delay_6us();
    SetPort_1Wire(input);
    Delay_9us();
      Dbit = Port_1Wire;
    Delay_10us(5);

      return Dbit;
}
//#########################################################################
################################
char iButton_ReadByte(void)
{
    char k = 0;
    char Dbyte = 0;

    for (k = 0; k < 8; k++)
        {
            Dbyte = Dbyte >> 1;
```

```
            if (iButton_ReadBit()) Dbyte |= 0x80;
        }

    return Dbyte;
}
//#########################################################################
################################
void iButton_WriteBit(char Dbit)
{
    if(Dbit)
    {
            Port_1Wire=0;
        SetPort_1Wire(output);
        Delay_6us();
        SetPort_1Wire(input);
        Delay_10us(5);
    }
    else
    {
            Port_1Wire=0;
        SetPort_1Wire(output);
        Delay_10us(5);
        SetPort_1Wire(input);
        Delay_6us();
    }

    Delay_9us();
}
//#########################################################################
################################
void iButton_WriteByte(char Dbyte)
{
    char k = 0;

      for (k = 0; k < 8; k++)
            {
            iButton_WriteBit(Dbyte & 0x01);
            Dbyte = Dbyte >> 1;
            }
}
//#########################################################################
################################
void Vyhodnot_data_od_UART()
{
    if(Buffer_Rx_UART[0]=='M' && Buffer_Rx_UART[3]=='E'  )
//Pokud vysila Slave a je v poradku startovaci a ukoncovali znak najdi
shodu iButton a sepni rele
    {
        if(Buffer_Rx_UART[1]==ID_Slave)
//Pokud souhlasi adresa Slave, provede prikaz
        {
            if(Buffer_Rx_UART[2]==0xF0) { LEDR=on; }
//Prikaz od Master - Ibutton OK - rozsvit
            if(Buffer_Rx_UART[2]==0xF1) { LEDG=on; }
//Prikaz od Master
            Delay(100);
        }
    }
```

```
}
//#########################################################################
##################################
//#########################################################################
##################################
//#########################################################################
##################################

void main (void)
{
      nastaveni_procesoru();
//---------------------------------------- NASTAVENI --------------------
--------

      Nastaveni_UARTU();
    Rizeni_toku=prijem;
      Restart_UART();
      Vymaz_Buffer_Rx_UART();

      GIE=1;
      PEIE=1;

    LEDB=on;            //Po resetu bliknou LED - kontrola kontaktniho pole
    LEDG=on;
    LEDR=on;

    Delay(100);

    LEDB=off;
    LEDG=off;
    LEDR=off;

    Flag_blokace=0;



//---------------------------------------- HLAVNI PROVOZNI SMYCKA -------
------------------

      do{

            CLRWDT();
            Rizeni_toku=prijem;
            LEDR=off;
            LEDG=off;
//----------------- Detekce a cteni iButton ------------------------------
------------------
           if(iButton_Reset() && !Flag_blokace)
            {
                Rizeni_toku=vysilani;                   //Prepnuti RS485 do
Tx
                LEDB=on;

                    iButton_WriteByte(0x33);            // Send read
command

                    for (int k = 0; k < 9; k++)         //Read iButton to
buffer
                    {Buffer_iButton[k] = iButton_ReadByte();}
```

```
//-------------------- Odelsnai paketu na Master -------------------------
-----------------
                    PosliZnak('S');                     //Start
                    PosliZnak(ID_Slave);                //ID cislo Slave
                    PosliZnak(Buffer_iButton[0]);       //Nasleduje cely
kod iButton (Rodina, Seriove cislo, CRC)
                    PosliZnak(Buffer_iButton[1]);
                    PosliZnak(Buffer_iButton[2]);
                    PosliZnak(Buffer_iButton[3]);
                    PosliZnak(Buffer_iButton[4]);
                    PosliZnak(Buffer_iButton[5]);
                    PosliZnak(Buffer_iButton[6]);
                    PosliZnak(Buffer_iButton[7]);
                    PosliZnak('E');                     //Konec

                    Delay(1);                           //Prodleva po
vyslani dat pred prepnuti linky na prijem (nutne)
                    Rizeni_toku=prijem;                 //Prepnuti RS485 do
Rx

                    Flag_blokace=1;                     //Po nacteni
iButton je dalsi cteni zablokovano na dobu cca 2 vteriny (ceka se na
zpracovani dat na strane Master a potvrzeni)
                    Casovac_blokace=100;


            }
            else
            {
                if(!Flag_blokace)
                {
                    LEDB=off;                                   //Neni-li
prilozen iButton, blika LED v kratkych intervalech (zarizeni pripraveno
cist))
                    SetPort_1Wire(input);
                    Delay(100);
                    LEDB=on;
                    Delay(1);
                    LEDB=off;
                }
            }
//-------------------- Kontrola prijmu dat od Master ----------------------
-------------------
            if(Flag_Rx_UART) { Vyhodnot_data_od_UART(); Restart_UART();
Flag_Rx_UART=0;}
//-------------------- Casovac ukonceni blokace po prilozeni iButton -------
-------------------
            if(--Casovac_blokace==0) { Flag_blokace=0; }

//----------------------

         }while(1);


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$

}
```

SEZNAM PŘÍLOH

Zdrojové soubory pro hlavní mikrokontroler

iButton_master.c,

iButton_master_iButtons.c,

iButton_master.h

iButton_master_retezce.h

Zdrojové soubory pro hlavní mikrokontroler čtečky

iButton_slave.c,

iButton_slave.h

iButton_slave_retezce.h

```
//#############################################################################################
//#############################################################################################
/*

    iButton - MASTER
```

```
                PIC18LF26K22
    IRC8 MHz
                1. UART (RS485)
    2. UART to PC
    1 x LED
                ....


                18.6.2018

*/
//#############################################################################################
#
//#############################################################################################
#
#include "htc.h"
#include "iButton_master.h"
#include "iButton_master_retezce.h"
#include "iButton_master_iButtons.h"
#include <xc.h>
//#############################################################################################
########
// CONFIG1H
#pragma config FOSC = INTIO67   // Oscillator Selection bits (Internal oscillator block)
#pragma config PLLCFG = OFF     // 4X PLL Enable (Oscillator used directly)
#pragma config PRICLKEN = ON    // Primary clock enable bit (Primary clock enabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover bit (Oscillator Switchover mode
disabled)

// CONFIG2L
#pragma config PWRTEN = OFF     // Power-up Timer Enable bit (Power up timer disabled)
#pragma config BOREN = SBORDIS  // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware
only (SBOREN is disabled))
#pragma config BORV = 190       // Brown Out Reset Voltage bits (VBOR set to 1.90 V nominal)

// CONFIG2H
#pragma config WDTEN = OFF      // Watchdog Timer Enable bits (Watch dog timer is always disabled.
SWDTEN has no effect.)
#pragma config WDTPS = 32768    // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H
#pragma config CCP2MX = PORTC1  // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF     // PORTB A/D Enable bit (PORTB<5:0> pins are configured as digital I/O
on Reset)
#pragma config CCP3MX = PORTB5  // P3A/CCP3 Mux bit (P3A/CCP3 input/output is multiplexed with RB5)
#pragma config HFOFST = OFF     // HFINTOSC Fast Start-up (HFINTOSC output and ready status are
delayed by the oscillator stable status)
#pragma config T3CMX = PORTC0   // Timer3 Clock input mux bit (T3CKI is on RC0)
#pragma config P2BMX = PORTB5   // ECCP2 B output mux bit (P2B is on RB5)
#pragma config MCLRE = INTMCLR  // MCLR Pin Enable bit (RE3 input pin enabled; MCLR disabled)

// CONFIG4L
#pragma config STVREN = OFF     // Stack Full/Underflow Reset Enable bit (Stack full/underflow will not cause
Reset)
#pragma config LVP = OFF        // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
```

```
#pragma config XINST = OFF      // Extended Instruction Set Enable bit (Instruction set extension and Indexed
Addressing mode disabled (Legacy mode))

// CONFIG5L
#pragma config CP0 = OFF        // Code Protection Block 0 (Block 0 (000800-003FFFh) not code-protected)
#pragma config CP1 = OFF        // Code Protection Block 1 (Block 1 (004000-007FFFh) not code-protected)
#pragma config CP2 = OFF        // Code Protection Block 2 (Block 2 (008000-00BFFFh) not code-protected)
#pragma config CP3 = OFF        // Code Protection Block 3 (Block 3 (00C000-00FFFFh) not code-protected)

// CONFIG5H
#pragma config CPB = OFF        // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-
protected)
#pragma config CPD = OFF        // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection Block 0 (Block 0 (000800-003FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection Block 1 (Block 1 (004000-007FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection Block 2 (Block 2 (008000-00BFFFh) not write-protected)
#pragma config WRT3 = OFF       // Write Protection Block 3 (Block 3 (00C000-00FFFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection bit (Configuration registers
(300000-3000FFh) not write-protected)
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot Block (000000-0007FFh) not write-
protected)
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF      // Table Read Protection Block 0 (Block 0 (000800-003FFFh) not protected
from table reads executed in other blocks)
#pragma config EBTR1 = OFF      // Table Read Protection Block 1 (Block 1 (004000-007FFFh) not protected
from table reads executed in other blocks)
#pragma config EBTR2 = OFF      // Table Read Protection Block 2 (Block 2 (008000-00BFFFh) not protected
from table reads executed in other blocks)
#pragma config EBTR3 = OFF      // Table Read Protection Block 3 (Block 3 (00C000-00FFFFh) not protected
from table reads executed in other blocks)

// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot Block (000000-0007FFh) not
protected from table reads executed in other blocks)

//#################################################################################################
#########
void Delay(char cas);


//#################################################################################################
#################
//#################################################################################################
################
//#################################################################################################
################
__interrupt()

void preruseni (void)
  {

      unsigned char ukazatel;
```

```
    unsigned short timer_Rx;

//------------------------- Preruseni od UARTU 1 (PC)
            if(RC1IF)
                    {
            ukazatel=0;
            timer_Rx=0;

            Flag_Rx_UART1=1;

                        do{
                                    if(RC1IF)
                                            {
                                                    Buffer_Rx_UART1[ukazatel]=RCREG1;
                                                    if(ukazatel<101) ++ukazatel;
                                                    timer_Rx=0;
                                            }

                                }while(++timer_Rx<250);

            while(RC1IF) { RC1IF=0; Registr_temp=RCREG1;}

                    }

//------------------------- Preruseni od UARTU 2 (RS485)
            if(RC2IF)
                    {
                        ukazatel=0;
                        timer_Rx=0;

                        Flag_Rx_UART2=1;

                        do{
                                    if(RC2IF)
                                            {
                                                    Buffer_Rx_UART2[ukazatel]=RCREG2;
                                                    if(ukazatel<101) ++ukazatel;
                                                    timer_Rx=0;
                                            }

                                }while(++timer_Rx<250);

            while(RC2IF) { RC2IF=0; Registr_temp=RCREG1;}

                    }



 }
//#########################################################################################
#################
//#########################################################################################
#################
void nastaveni_procesoru()
{

   OSCCON  =  0B01100010;     //8MHz
```

```
   OSCCON2 =  0B00000111;
//
         PORTA            =  0xFF;
         PORTB   =  0xFF;
   PORTC         =  0xFF;
//
   TRISA         =  0B00000000;
   TRISB   =  0B10000000;
   TRISC         =  0B10000000;

//
   WDTCON  =  0B00011001;

   INTCON  =  0x00;
   INTCON2 =  0x00;
   INTCON3 =  0x00;
   PIR1    =  0x00;
   PIR2    =  0x00;
   PIR3    =  0x00;
   PIR4    =  0x00;
   PIR5    =  0x00;

         PIE1    = 0x00;
         PIE2    = 0x00;
         PIE3    = 0x00;
         PIE4    = 0x00;
         PIE5    = 0x00;

   PMD0    = 0xFF;
   PMD1    = 0xFF;
   PMD2    = 0xFF;



   ANSELA  =  0x00;
   ANSELB  =  0x00;
   ANSELC  =  0x00;

   WPUB    =  0;

   FVRCON  =  0x00;

   ADCON0  =  0x00;
   ADCON1  =  0x00;

   CM1CON0 =  0x00;
   CM2CON0 =  0x00;

         CCP1CON = 0;
   CCP2CON = 0;
   CCP5CON = 0;

         SSP1CON1= 0;
         SSP1CON3= 0;
   SSP2CON1= 0;
   SSP2CON3= 0;
```

```
        TMR1ON          = 0;
        TMR2ON  = 0;
   T2CON   = 0;

   ECCP1AS = 0x00;
   ECCP1AS = 0x00;

   CTMUCONH = 0x00;
   SLRCON   = 0x00;



}
//#############################################################################################
#################
void Nastaveni_UART1()
{
//---------- Nastaveni USART1 - PC

        TXSTA1          = 0B00100100;     //OSC8MHz / 9600B
        RCSTA1          = 0B10000000;
        BAUDCON1 = 0B00001000;
        SPBRG1          = 207;
        SPBRGH1         = 0;
        while(RC1IF) { RC1IF=0; NOP(); Registr_temp=RCREG1;}
   SPEN1    = 1;
   CREN1    = 1;
   RC1IE    = 1;
   UART1MD  = 0;
//Povoleni preruseni
   GIE_GIEH  = 1;
   PEIE_GIEL = 1;
}
//#############################################################################################
#################
void Nastaveni_UART2()
{

//---------- Nastaveni USART2 - RS485

        TXSTA2          = 0B00100100;     //OSC8MHz / 9600B
        RCSTA2          = 0B10000000;
        BAUDCON2 = 0B00001000;
        SPBRG2          = 207;
        SPBRGH2         = 0;
        while(RC2IF) { RC2IF=0; NOP(); Registr_temp=RCREG;}
   SPEN2    = 1;
   CREN2    = 1;
   RC2IE    = 1;
   UART2MD  = 0;
//Povoleni preruseni
   GIE_GIEH  = 1;
   PEIE_GIEL = 1;
}
//#############################################################################################
#################
```

```
//#############################################################################################
#################
void Delay(char cas)
{
   unsigned int  e=0;

   do {  e++;

                              {
                                       char k;
                                       for (k=cas;k>1;k--)          continue;
                              }
            } while (e<589);
}
//#############################################################################################
#################
//#############################################################################################
#################
void PosliZnak(char UARTx, char znak)
{
   if(UARTx)
   {
      //-------- PC
      while(!TX1IF);
      TXREG1 = znak; NOP();NOP();NOP();NOP();NOP();
      while(!TX1IF);
   }
   else
   {
      //-------- RS485
      while(!TX2IF);
      TXREG2 = znak; NOP();NOP();NOP();NOP();NOP();
      while(!TX2IF);
   }
}
//*********************************************************************************************
*****************
void PosliText (char UARTx, char *s) // posila retezce na UART
{
          do {
      char znak = *s++;
      if (!znak) {break; }
      if(UARTx) { TX1REG = znak; } else { TX2REG = znak; }
      NOP(); NOP(); NOP();
      if(UARTx) { while(!TX1IF) {NOP(); }; } else while(!TX2IF) {NOP(); };
     } while (1);
}
//#############################################################################################
#################
void prevod_HEX_Ascii(char vstup_hex)
{
   MSB=vstup_hex>>4; LSB=vstup_hex&0x0F;
   if(MSB>9) { MSB=MSB+55; } else { MSB=MSB+48; }
   if(LSB>9) { LSB=LSB+55; } else { LSB=LSB+48; }
}
//#############################################################################################
#################
void Vymaz_Buffer_Rx_UART1()
```

```
{
        for(char i=0;i<101;i++) Buffer_Rx_UART1[i]=0xFF;
}
//##############################################################################################
#################
void Vymaz_Buffer_Rx_UART2()
{
        for(char i=0;i<101;i++) Buffer_Rx_UART2[i]=0xFF;
}
//##############################################################################################
#################
void Restart_UART1()
{
        Flag_Rx_UART1=0;
        Nastaveni_UART1();
        Vymaz_Buffer_Rx_UART1();
}
//##############################################################################################
#################
void Restart_UART2()
{
        Flag_Rx_UART2=0;
        Nastaveni_UART2();
        Vymaz_Buffer_Rx_UART2();
}
//##############################################################################################
#################
void Vyhodnot_vystup(char Flag_opravneni)
{
  if(Flag_opravneni==ano)
  {
    LEDG=on;                    //opravneni OK, sepnuti RELE (zelena LED)
  //-------- Nasleduje odeslani informace do PC: --------------------
    PosliText(PC,(char*)" >>>> ");     //uvodni znaky

    PosliText(PC,(char*)S_Kod_Slave);    //kod ctecky
    prevod_HEX_Ascii(Buffer_Rx_UART2[1]);
    PosliZnak(PC,MSB); PosliZnak(PC,LSB); //... odeslan na PC v ASCII

    PosliText(PC,(char*)S_Kod_iButton);  //kod iButton

    char pozice=2;
    do{                         //odesilani kodu iButton v ASCII formatu
      prevod_HEX_Ascii(Buffer_Rx_UART2[pozice]);
      PosliZnak(PC,MSB); PosliZnak(PC,LSB); PosliZnak(PC,' ');
    }while(++pozice<10);

    PosliText(PC,(char*)S_Opravneni);    //Opraveni ANO
    PosliText(PC,(char*)S_Ano);

    PosliText(PC,(char*)" <<<< ");     //ukoncovaci znaky
  }
  else
  {
    LEDR=on;                    //opravneni NOK, nezapnuti RELE (zelena LED)
  //-------- Nasleduje odeslani informace do PC: --------------------
    PosliText(PC,(char*)" >>>> ");     //uvodni znaky
```

```
      PosliText(PC,(char*)S_Kod_Slave);     //kod ctecky ...
      prevod_HEX_Ascii(Buffer_Rx_UART2[1]);
      PosliZnak(PC,MSB); PosliZnak(PC,LSB); //... odeslan na PC v ASCII


      PosliText(PC,(char*)S_Kod_iButton);   //kod iButton

      char pozice=2;
      do{                          //odesilani kodu iButton v ASCII formatu
        prevod_HEX_Ascii(Buffer_Rx_UART2[pozice]);
        PosliZnak(PC,MSB); PosliZnak(PC,LSB); PosliZnak(PC,' ');
      }while(++pozice<10);

      PosliText(PC,(char*)S_Opravneni);     //Opraveni NE
      PosliText(PC,(char*)S_Ne);

      PosliText(PC,(char*)" <<<< ");      //ukoncovaci znaky
   }
}
//#############################################################################################
#################
bit Porovnej_kody_iButton()
{
  char Buff_pointer=2;
  char Page_pointer=0;

      do{
        if(Buffer_Rx_UART2[Buff_pointer]==iButtons_code_array[(Buff_pointer-2)+Page_pointer])
//Prohledava pole a porovnava s prijatym kodem iButton
            {
              if(Buff_pointer==9) { return 1; } else { ++Buff_pointer; }
            }
            else
            {
              Page_pointer=Page_pointer+8; Buff_pointer=2;
            }

        }while(Page_pointer<=72);

   return 0;
}
//#############################################################################################
#################
void Vyhodnot_data_od_UART1()
{

}
//#############################################################################################
#################
void Vyhodnot_data_od_UART2()
{
   Flag_shoda=0;

   if(Buffer_Rx_UART2[0]=='S' && Buffer_Rx_UART2[10]=='E'  )          //Pokud vysila Slave a je v poradku
startovaci a ukoncovali znak najdi shodu iButton a sepni rele
   {
      ID_Slave=Buffer_Rx_UART2[1];                        //Tady je ulozeno cislo ctecky
```

```
      if(Porovnej_kody_iButton()==nalezena_shoda) Flag_shoda=1;      //Porovna se prijate ID iButton s temi,
ulozenymi v pameti
      Vyhodnot_vystup(Flag_shoda);
      Flag_odeslat_odpoved=1;
   }
}
//#############################################################################################
#################
void Odeslat_odpoved(char ID_Slave, char Flag_shoda)
{
   Rizeni_toku=vysilani;
   Delay(10);

   PosliZnak(RS485,'M');                              //Startovaci znak od Master je 'M'
   PosliZnak(RS485,ID_Slave);                          //Adresat
   if(Flag_shoda)
   {
      PosliZnak(RS485,0xF1);                           //Prikaz (0xF0=rozsvit cervenou 0xF1=rozsvit zelenou
LED)
   }
   else
   {
      PosliZnak(RS485,0xF0);                           //Prikaz (0xF0=rozsvit cervenou 0xF1=rozsvit zelenou
LED)
   }
   PosliZnak(RS485,'E');                              //Ukoncovaci znak

   Delay(1);
   Rizeni_toku=prijem;
}
//#############################################################################################
#################
//#############################################################################################
#################

void main (void)
{
         nastaveni_procesoru();
   Rizeni_toku=prijem;

//------------------------------------------ NASTAVENI ----------------------------

         Nastaveni_UART1();
   Nastaveni_UART2();

         Restart_UART1();
   Restart_UART2();

         Vymaz_Buffer_Rx_UART1();
   Vymaz_Buffer_Rx_UART2();

   GIE=1;          //Globalni povoleni preruseni (pouzivame od Uartu))
   PEIE=1;

   LEDG=off;
   LEDR=off;
   LEDY=off;
```

```
 //------------------------------------------ HLAVNI PROVOZNI SMYCKA ----------------------------------
            do{
      Flag_odeslat_odpoved=0;
      Rizeni_toku=prijem;
//------------ Zluta, stavova LED ------------------------------------------------------------------
      LEDY=!LEDY; //Stavova LED. Pokud je zarizeni pripraveno, blika.
//------------ Vypnuti ostatnich LED ---------------------------------------------------------------
      LEDG=off; LEDR=off;
//------------ Jen prodleva definujici rychlost blikani zlute LED ----------------------------------
      Delay(50);
//------------ Vyhodnoceni dat od  PC --------------------------------------------------------------
                  //if(Flag_Rx_UART1) { Vyhodnot_data_od_UART1(); Restart_UART1(); Flag_Rx_UART1=0;
}
//------------ Vyhodnoceni dat od RS485 (iButton) --------------------------------------------------
      if(Flag_Rx_UART2) { LEDY=on; Vyhodnot_data_od_UART2(); Restart_UART2(); Flag_Rx_UART2=0;
}
//------------ Pokud byl prijem od Slave uspesny, je odeslano potvrzeni ----------------------------
//------------ Pak je jeste pridana delsi prodleva definujici svit cervene/zelene LED --------------
      if(Flag_odeslat_odpoved) { Odeslat_odpoved(ID_Slave,Flag_shoda); Flag_odeslat_odpoved=0;
Delay(100); }
//----------- Nulovani WDT -------------------------------------------------------------------------
                  CLRWDT();
//---------- Konec hlavni smycky -------------------------------------------------------------------

            }while(1);


//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$

}
```